«a2» 2013.11.11 --- Copyright © Daniele Giacomini -- appunti2@gmail.com http://informaticalibera.net

X è un sistema grafico per gli ambienti Unix, o più precisamente per gli ambienti aderenti agli standard C ANSI o POSIX. X Window System è stato sviluppato originariamente nei laboratori del MIT (*Massachusetts institute of technology*) e in seguito tutti i diritti sono stati assegnati a un altro ente.

I termini X, X Window e X Window System sono da intendersi come sinonimi dello stesso sistema grafico, mentre il nome «X Windows» non è corretto.

X Window System è un marchio registrato. Lo sviluppo di X come software libero avviene attraverso la fondazione X.Org (*http://www.x.org*).

28.1 Struttura

Nel sistema X si utilizzano alcuni termini importanti che rappresentano altrettante parti di questo.

- **servente X**

  Il servente X è il programma che gestisce le funzionalità grafiche e le mette a disposizione degli altri programmi. Per questa ragione, l'elaboratore su cui si fa funzionare il servente X deve essere dotato di video grafico, tastiera e mouse. Il servente grafico fornisce anche un servizio di rete dal momento che consente l'accesso a programmi in funzione presso altri elaboratori.

- **cliente X**

  I clienti X sono i programmi che utilizzano questo ambiente grafico comunicando con il servente X. Un cliente X può essere messo in funzione anche in un elaboratore diverso da quello sul quale è in funzione un servente X.

- **protocollo X**

  Tra i clienti X e il servente X, intercorre una comunicazione, attraverso un protocollo prestabilito.

- **librerie**

  I programmi che utilizzano i servizi del servente X accedono a funzioni offerte da librerie apposite: quelle fondamentali sono Xlib e XCB.

- **gestore di finestre**

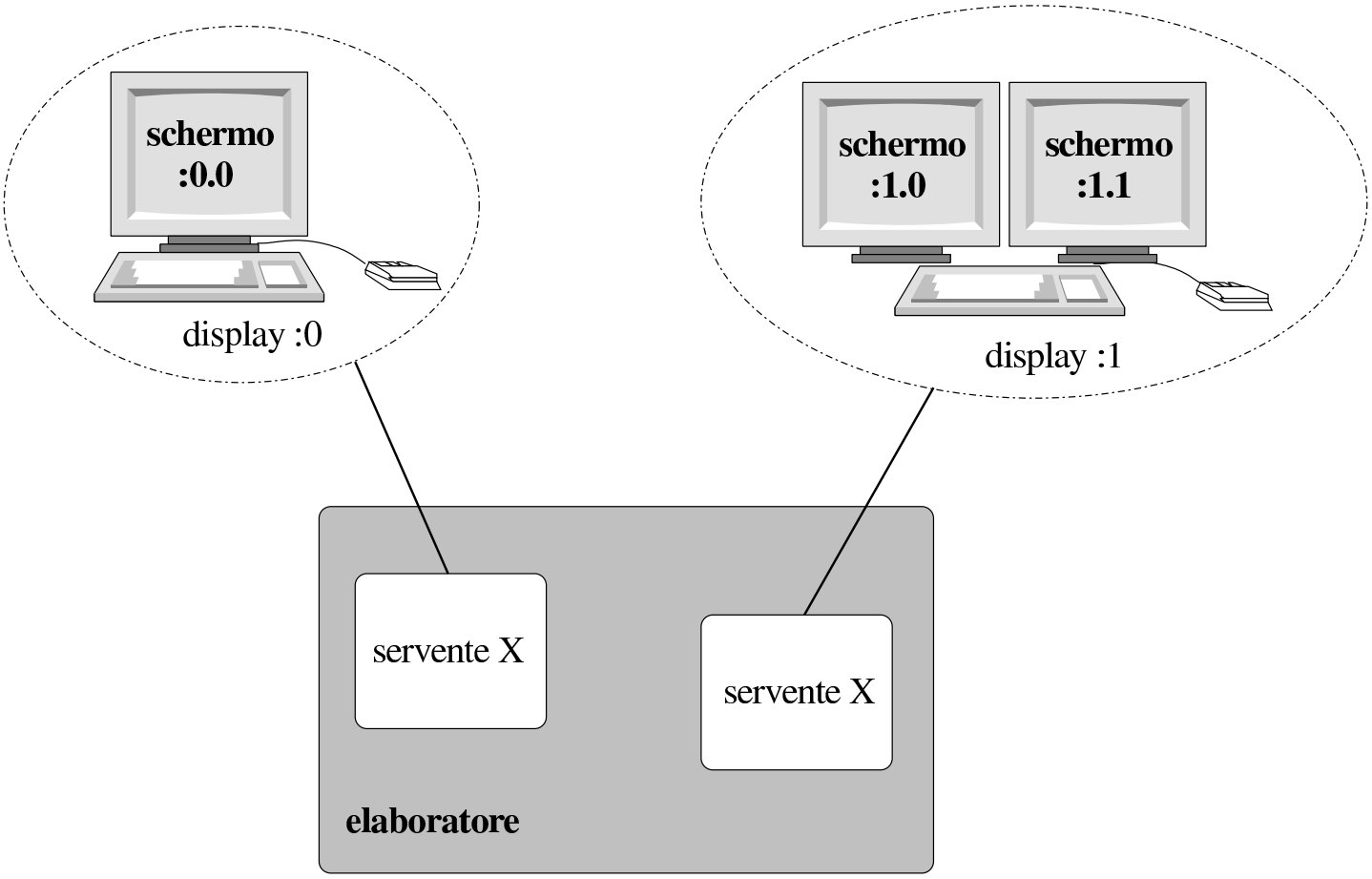
  Un gestore di finestre, ovvero un *window manager*, è un programma speciale che si occupa di gestire le finestre delle varie applicazioni. In generale, nell'ambiente X si tratta di un cliente X.

28.1.1 Hardware

«

Dal punto di vista di X, l'hardware è ciò che consente di interagire in questo sistema grafico (nel senso che il resto non è di sua competenza). Si tratta della tastiera, dello schermo grafico e del dispositivo di puntamento. In pratica il ruolo di X è quello di controllare tutto questo.

Figura 28.1. X è un sistema attraverso il quale, teoricamente, è possibile avere macchine che fanno girare più di un servente grafico, ognuno in grado di controllare una stazione grafica (*display*) che a sua volta utilizza uno o più schermi grafici.



All'interno di un elaboratore possono funzionare teoricamente più serventi grafici per controllare altrettante stazioni grafiche di lavoro. Inoltre, sempre teoricamente, una stazione grafica può utilizzare più di uno schermo grafico contemporaneamente.

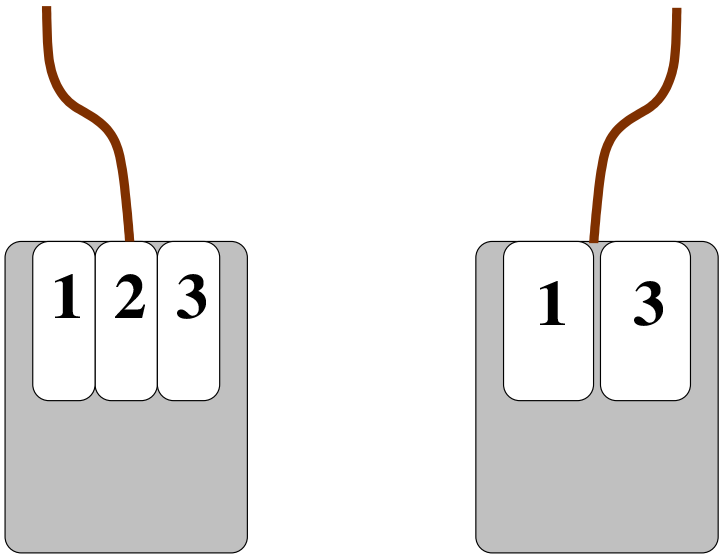
Nel gergo di X la stazione grafica è il *display*, che viene identificato da un numero a partire da zero, nella forma '**:*n***'. Se una stazione grafica è dotata di più di uno schermo, quando si deve fare riferimento a uno di questi occorre aggiungere all'indicazione del numero della stazione grafica quello dello schermo. Anche in questo caso, il pri-

mo corrisponde a zero. La forma diventa quindi ‘`:`***n*** `.`***m***’, dove ***n*** è la stazione grafica e ***m*** è lo schermo. La figura 28.1 dovrebbe chiarire il meccanismo. Il valore predefinito di stazione grafica e schermo è zero, per cui, quando non si specificano queste informazioni, si intende implicitamente lo schermo ‘`:0.0`’.

I dispositivi di puntamento, solitamente il mouse, possono avere un numero variabile di tasti; teoricamente si va da un minimo di uno a un massimo di cinque. Nell’ambiente X, questi tasti si distinguono attraverso un numero: 1, 2, 3, 4 e 5. Il tasto sinistro è il primo e da lì si continua la numerazione. Quando si utilizza un mouse a tre tasti, il tasto numero due è quello centrale.

Per X è necessario disporre di almeno tre tasti: nei mouse a due tasti, il tasto destro svolge la funzione del tasto numero tre e solitamente il tasto centrale (cioè il numero due) si ottiene con l’uso contemporaneo dei due tasti esistenti.

Figura 28.2. La numerazione dei tasti dei mouse che ne hanno solo due è particolare.



Questo problema viene ripreso nella descrizione della configurazio-

ne di X e lì dovrebbe risultare più chiaro.

28.1.2 Cliente-servente

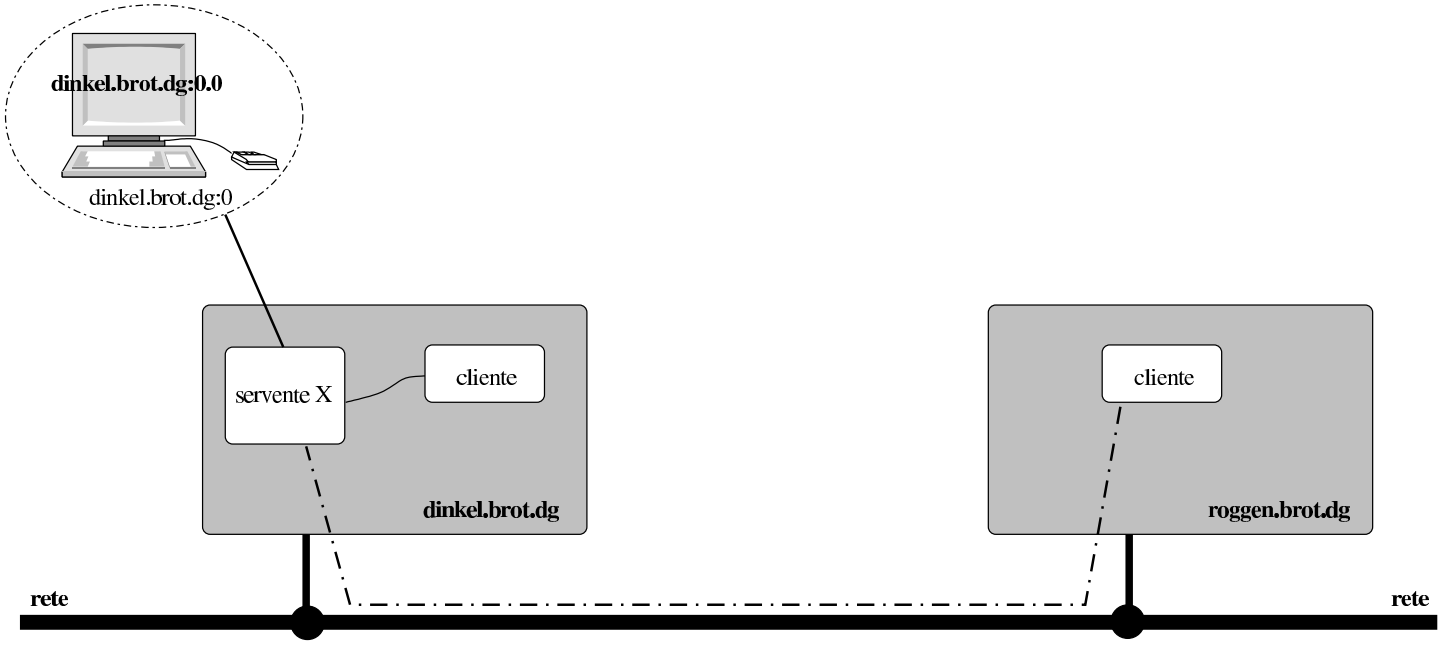
Il programma che si occupa di gestire la stazione grafica è il servente grafico. È un servente perché offre solo dei servizi e non interagisce direttamente con l'utente. Sono i programmi clienti a interagire con l'utente. Questi richiedono al servente di poter utilizzare uno schermo determinato e, attraverso la stazione grafica corrispondente, sono in grado di ricevere l'input della tastiera e del dispositivo di puntamento.

Tra i programmi clienti, quello che riveste un ruolo fondamentale è il gestore di finestre, attraverso il quale si rendono disponibili quei meccanismi con cui si può passare facilmente da un programma all'altro e le finestre possono essere ridimensionate o ridotte a icona.

X è trasparente nei confronti della rete. Un programma cliente può utilizzare i servizi di un servente remoto, interagendo con la stazione grafica di quel servente. Questo tipo di utilizzo richiede comunque una forma di autorizzazione o autenticazione, per motivi di sicurezza.

Quando si vuole identificare uno schermo particolare di un certo elaboratore nella rete, si antepone alle coordinate (già viste nella sezione precedente) il nome o l'indirizzo di quell'elaboratore: '***nodo*:*n*.*m***'. La figura 28.3 mostra un esempio di questo tipo di utilizzo.

Figura 28.3. Il servente grafico può concedere l'utilizzo della stazione grafica anche a programmi in esecuzione su elaboratori remoti.



Un programma cliente può connettersi con un servente X, sia locale, sia remoto. Per una connessione remota occorre stabilire un collegamento. Il servente X resta normalmente in ascolto sulla porta 6000 + ***n***, dove ***n*** rappresenta il numero della stazione grafica, ovvero del servente X.

Nel caso di una stazione grafica con indirizzo '`:1`', la porta su cui dovrebbe trovarsi in ascolto il servente relativo è la numero 6001.

Il concetto di cliente-servente per ciò che riguarda la rete viene ripreso nei capitoli dedicati proprio alle connessioni in rete (capitolo 32 e successivi).

Infine, è bene tenere in considerazione il fatto che le librerie specifiche per X (Xlib), sono indispensabili sia per i serventi, sia per i clienti.

28.2 X.Org

La versione libera di X è costituita da X.Org[1] e in precedenza è stata XFree86.[2] In generale si tratta di un servente X per i sistemi operativi Unix, eventualmente in più versioni alternative, specifiche per un gruppo particolare di adattatori grafici.

La struttura originaria prevista per il file system di un sistema GNU e di altri sistemi Unix collocava tutti i file statici di X (binari, documentazione, librerie, ecc.) al di sotto di '`/usr/X11R6/`', ma attualmente questa anomalia sta scomparendo.

X utilizza un file di configurazione, che può essere collocato in varie posizioni, ma generalmente si trova nella directory '`/etc/X11/`'. Il file di configurazione può avere nomi diversi in base alla realizzazione di X, ma nel caso di X.Org si tratta semplicemente di '`xorg.conf`'. In caso di necessità può essere generato un file di configurazione iniziale con il comando '**`X -configure`**':

```
# X -configure [Invio]
```

Ciò che si ottiene il file '`xorg.conf.new`' nella directory personale dell'utente '**`root`**'. Questo file potrebbe essere usato per rimpiazzare '`/etc/X11/xorg.conf`'.

28.3 Lettura del file di configurazione

La lettura del file di configurazione di X ('`/etc/X11/xorg.conf`') può dare molte informazioni utili sull'organizzazione del sistema grafico. In particolare, i programmi utilizzati per generarlo sono realizzati in modo da inserire molti commenti, tra cui anche esempi di direttive, così da agevolare chi volesse modificarlo successivamente a mano.

A titolo di esempio viene illustrato un file di configurazione generalizzato, suddividendolo nelle sue sezioni, adatto alla maggior parte dell'hardware comune, aderente alle specifiche VESA.

Nel file, il simbolo '**#**' serve a iniziare un commento che termina alla fine della riga e le righe bianche o vuote vengono ignorate; inoltre, le direttive che occupano più righe vengono separate semplicemente, senza bisogno si simboli di continuazione. Le direttive del file sono raggruppate in sezioni, dichiarate nel modo seguente:

```
Section "nome_sezione"
    direttiva
    direttiva
    ...
EndSection
```

All'interno del file possono essere definiti vari schermi, tastiere e mouse, senza necessariamente che tutto debba essere utilizzato o utilizzabile. Per riassumere ciò che serve e che deve essere attivato nella gestione di X, si utilizza la sezione '**ServerLayout**'. In questo caso si definisce l'impostazione '**Layout0**' che mette assieme lo schermo '**Screen0**', la tastiera '**Keyboard0**' e il mouse '**Mouse0**':

```
Section "ServerLayout"
        Identifier      "Layout0"
        Screen      0   "Screen0"      0 0
        InputDevice     "Keyboard0"    "CoreKeyboard"
        InputDevice     "Mouse0"       "CorePointer"
EndSection
```

La sezione '**Files**' serve a determinare la collocazione dei file usati da X. In questo caso si dichiara la collocazione dei moduli di X e dei

file che descrivono i caratteri tipografici:

```
Section "Files"
        ModulePath  "/usr/lib/xorg/modules"
        FontPath    "/usr/share/fonts/X11/misc"
        FontPath    "/usr/share/fonts/X11/cyrillic"
        FontPath    "/usr/share/fonts/X11/100dpi/:unscaled"
        FontPath    "/usr/share/fonts/X11/75dpi/:unscaled"
        FontPath    "/usr/share/fonts/X11/Type1"
        FontPath    "/usr/share/fonts/X11/100dpi"
        FontPath    "/usr/share/fonts/X11/75dpi"
        FontPath    "/usr/share/fonts/truetype/"
        FontPath    "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
        FontPath    "/var/lib/defoma/x-ttcidfont-font.d/dirs/CID/"
        FontPath    "unix/:7100"        # local font server
        FontPath    "built-ins"
EndSection
```

I moduli che si vuole siano usati da X sono elencati nella sezione '**Module**'. Eventualmente possono essere indicati anche moduli che in pratica non sono presenti: nel caso non vengono caricati.

```
Section "Module"
        Load    "dri2"
        Load    "bitmap"
        Load    "dbe"
        Load    "ddc"
        Load    "dri"
        Load    "extmod"
        Load    "freetype"
        Load    "glx"
        Load    "int10"
        Load    "record"
        Load    "type1"
        Load    "v4l"
        Load    "vbe"
EndSection
```

Nelle sezioni '**InputDevice**' si dichiarano principalmente la tastiera e il mouse (o dei componenti equivalenti). In questo caso la sezione viene usata per la tastiera e si attribuiscono anche opzioni speciali per garantire che funzionino i livelli terzo e quarto, ottenuti con la combinazione del tasto [*Maiuscole*]; inoltre si richiede espressamente che la combinazione [*Ctrl Alt Backspace*] chiuda il funzionamento di X. Per garantire il funzionamento corretto di quanto configurato, ci si può anche avvalere del comando '**setxkbmap**', così come suggerito dai commenti inseriti in questa sezione:

```
Section "InputDevice"
        Identifier      "Keyboard0"
        Driver          "kbd"
        Option          "CoreKeyboard"
        #
        Option          "AutoRepeat"    "500 30"
        Option          "LeftAlt"       "Meta"
        Option          "RightAlt"      "ModeShift"
        #
        # setxkbmap -rules xorg -model pc105 -layout "it,it" \
        #           -option "" -option "grp:alt_shift_toggle" \
        #           -option "terminate:ctrl_alt_bksp"
        # setxkbmap -rules evdev
        #
        Option          "XkbRules"      "xorg"
        Option          "XkbModel"      "pc105"
        Option          "XkbLayout"     "it,it"
        Option          "XkbOptions"
                        "grp:alt_shift_toggle,terminate:ctrl_alt_bksp"
EndSection
```

Un'altra sezione '**InputDevice**' si utilizza per il mouse, che in questo caso è controllato da un demone esterno, '**gpm**', il quale comunica le azioni del mouse attraverso il file di dispositivo '`/dev/gpmdata`' (14.10). Il mouse ha effettivamente solo tre tasti, dove quello centra-

le è costituito dalla rotellina. Tuttavia, qui si considera la presenza di cinque tasti, perché il movimento della rotellina, in un verso o nell'altro, assume il significato della pressione dei tasti mancanti.

```
Section "InputDevice"
        Identifier      "Mouse0"
        Driver          "mouse"
        Option          "CorePointer"
        Option          "Device"                "/dev/gpmdata"
        Option          "Protocol"              "IntelliMouse"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
        Option          "Buttons"               "5"
EndSection
```

Per definire l'adattatore grafico si usa la sezione '**`Device`**'. In questo caso si fa riferimento semplicemente a una compatibilità generica con lo standard VESA:

```
Section "Device"
        Identifier  "Card0"
        Driver      "vesa"
EndSection
```

La sezione '**`Monitor`**' descrive la modalità di funzionamento dello schermo, in particolare in merito alla frequenza di scansione orizzontale e verticale. Di solito, come in questo caso, si indicano gli intervalli di frequenze ammissibili, tenendo presente che la frequenza orizzontale è espressa in hertz (Hz), mentre quella verticale è espresso in kilohertz (kHz).

```
Section "Monitor"
        Identifier  "Monitor0"
        Option      "DPMS"
        HorizSync   30-80
        VertRefresh 50-70
EndSection
```

La sezione '**Screen**' permette di definire la risoluzione dello schermo, nella profondità di colori e nella geometria che dovrebbe avere. In questo caso, a proposito di geometria, vengono date diverse possibilità:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth    16
        SubSection "Display"
                Depth     16
                Modes     "1280x1024" "1152x870" "1024x768"
                          "832x624" "800x600" "720x400"
                          "640x480"
                ViewPort  0 0
        EndSubSection
EndSection
```

28.4 Accesso a un servente di caratteri

Nella sezione '**Files**' della configurazione di X si trovano in particolare le direttive '**FontPath**', per dichiarare la collocazione dei file contenenti le informazioni sui caratteri tipografici da visualizzare. Secondo questo tipo di impostazione, ogni volta che si aggiunge una directory contenente altri caratteri, occorre modificare la configurazione di X per includere anche quella tra i percorsi previsti. Per

semplificare l'accesso ai caratteri esistono dei ***serventi di caratteri***, con i quali X può comunicare attraverso la rete, oppure solo dei socket di dominio Unix. In altri termini, un servente di caratteri può offrire il suo servizio attraverso la rete, per più di un elaboratore, oppure anche solo localmente, per mezzo di file socket.

Esistono diversi programmi che possono svolgere il compito di un servente di caratteri (per esempio Xfs,[3] X-TT[4] e Xfstt[5]); inoltre, spesso l'installazione di un servente del genere diventa quasi obbligatoria per via delle dipendenze stabilite da chi organizza la propria distribuzione GNU. Senza entrare nell'analisi del funzionamento di un servente di caratteri, basti sapere che di solito questi sono in funzione in attesa di connessioni sulla porta 7100 TCP e se usano un socket di dominio Unix, dovrebbe corrispondere al file '`/tmp/.font-unix/fs7100`'. Se poi si gestiscono più serventi di caratteri nello stesso elaboratore, il numero della porta potrebbe essere un valore leggermente più alto, come 7101 o 7110, a cui si associa di conseguenza il file '`/tmp/.font-unix/fs7101`' o '`/tmp/.font-unix/fs7110`'.

In presenza di uno o più serventi di caratteri, si deve intervenire nella configurazione di X per dichiarare come questi possono essere raggiunti. In caso di socket di dominio Unix, si usano direttive di questo tipo:

```
FontPath "unix/:71nn"
```

In caso di connessioni attraverso la rete, si può provare una di queste due direttive:

```
FontPath "inet/nodo:71nn"
```

```
FontPath "tcp/nodo:71nn"
```

Naturalmente, ***nn*** va sostituito con il valore esatto, in base alla configurazione del servente di caratteri a cui si vuole accedere.

28.5 Funzionamento e accesso

«

Con le distribuzioni GNU normali, dopo la configurazione del servente X, dovrebbe essere sufficiente avviare lo script '**startx**', senza argomenti, per vedere funzionare questo ambiente grafico.

```
$ startx [Invio]
```

Avendo avviato il servente X, vale la pena di provare a cambiare la risoluzione di visualizzazione attraverso la combinazione [*Ctrl Alt* ⊞] («control», «alt», «+ del tastierino numerico») e [*Ctrl Alt* ⊟] («control», «alt», «– del tastierino numerico»).

Per passare dal servente X a una console virtuale, è sufficiente utilizzare la combinazione [*Ctrl Alt F1*], oppure [*Ctrl Alt F2*],... invece del solito [*Alt Fn*] che non potrebbe funzionare. Il servente X occupa normalmente la posizione della prima console virtuale libera, che solitamente è la settima; per cui si raggiunge con la combinazione [*Ctrl Alt F7*].

Per concludere l'esecuzione del servente X ci sono due modi:

- interrompere il servente attraverso la combinazione [*Ctrl Alt Backspace*];

- concludere l'esecuzione del gestore di finestre o di altro programma analogo.

L'interruzione dell'esecuzione del servente X con la combinazione [*Ctrl Alt Backspace*] è il modo più brutale, ma può essere opportuno quando non si vede più nulla, specie quando si è avviato X dopo una configurazione sbagliata. Tuttavia, in alcuni sistemi può essere disattivata questa opzione, indipendentemente dalla configurazione contenuta nel file '`/etc/X11/xorg.conf`'. Per assicurarsi di attivare questa opzione (ammesso che la si voglia), si può usare il comando seguente:

```
$ setxkbmap -option terminate:ctrl_alt_bksp [Invio]
```

28.5.1 Procedura di avvio

Nelle sezioni precedenti si è accennato al modo con cui è possibile avviare e concludere il funzionamento del servente X. Dovrebbe essere chiaro che per avviare X si utilizza normalmente lo script '**`startx`**' (anche se non è l'unico modo possibile), dal quale si sviluppa una struttura piuttosto articolata che è opportuno conoscere.

Quando sono disponibili diversi serventi grafici distinti a seconda del tipo di adattatore grafico, si crea un collegamento simbolico in modo da poter avviare il servente giusto utilizzando semplicemente il nome '**`X`**'. In pratica, dovrebbe essere il programma di configurazione stesso che provvede a sistemare questa cosa.

Se si avvia semplicemente il servente, utilizzando il nome '**`X`**' oppure quello specifico di un adattatore grafico particolare, si ottiene solo una superficie grafica su cui fare scorrere il mouse. Per poter

fare qualcosa, occorre almeno avere in funzione un programma che consenta di avviarne altri. Occorrono cioè dei clienti.[6]

Per risolvere questo problema si deve utilizzare il programma '**xinit**', attraverso il quale si possono definire alcuni clienti di partenza (per esempio un gestore di finestre), il tipo di servente da utilizzare e le sue opzioni eventuali.

28.5.1.1 Utilizzo di «xinit»

«

Il programma '**xinit**' viene usato per avviare il servente X e un primo programma cliente. Quando questo programma cliente termina la sua esecuzione, '**xinit**' invia un segnale di interruzione al servente X e quindi, a sua volta, termina la sua esecuzione.

```
xinit [[cliente] opzioni] [ -- [servente] [stazione_grafica] opzioni]
```

Se non viene indicato un programma cliente specifico, '**xinit**' tenta di avviare il file '`~/.xinitrc`', che di solito dovrebbe corrispondere a uno script; se questo manca, tenta di avviare il programma '**xterm**' nel modo seguente:

```
xterm -geometry +1+1 -n -login -display :0
```

Se non viene indicato un programma servente specifico, '**xinit**' tenta di avviare il file '`~/.xserverrc`'; se questo manca, tenta di avviare il programma '**X**' nel modo seguente:

```
X :0
```

Quando si vuole fare in modo che il servente X venga avviato inizialmente con un gruppetto di programmi clienti, si fa in modo che '**xinit**' utilizzi per questo uno script. Di solito si tratta proprio del file '`~/.xinitrc`', quello che verrebbe avviato in modo predefinito. All'interno di questo script, i programmi dovrebbero essere avviati sullo sfondo, con la possibile eccezione di quelli che terminano immediatamente la loro funzione. L'ultimo di questi programmi deve funzionare in primo piano (*foreground*), in modo che la sua conclusione corrisponda con quella dello script stesso.

Di solito, '**xinit**' viene avviato senza l'indicazione esplicita di cliente e servente. Se si intende utilizzare questa possibilità, i nomi di cliente e servente devono comprendere il percorso per raggiungerli: devono cioè iniziare con un punto ('**.**') oppure con una barra obliqua ('**/**'). Diversamente non verrebbero riconosciuti come tali, ma come opzioni per il programma cliente o per il programma servente, a seconda che si trovino a sinistra o a destra dei due trattini di separazione ('**--**'). Segue la descrizione di alcuni esempi.

- `$` **`xinit &`**[*Invio*]

  Avvia '**xinit**' con i valori predefiniti (sullo sfondo). In questo modo '**xinit**' tenta di avviare il servente X utilizzando il programma o lo script '`~/.xinitrc`' come cliente, oppure il programma '**xterm**' in sua mancanza.

- `$` **`xinit -- /usr/bin/X11/X &`**[*Invio*]

  Si richiede a '**xinit**' di avviare il servente '`/usr/bin/X11/X`' (probabilmente è un programma con privilegi speciali che a sua volta avvia '`/usr/bin/X11/Xorg`'). Per quanto riguarda il cliente, si utilizzano i valori predefiniti.

- `$` **`xinit -- -depth 16`** [*Invio*]

  ‘**`xinit`**’ avvia il servente X predefinito, con l’argomento ‘**`-depth 16`**’, attraverso cui si richiede una profondità di colori di 16 bit/pixel (2^{16} = 65 535). Per quanto riguarda il cliente, si utilizzano i valori predefiniti.

Il modo migliore per verificare cosa accade quando si avvia ‘**`xinit`**’ è quello di verificare l’interdipendenza tra i processi attraverso ‘**`pstree`**’. Supponendo di avere avviato ‘**`xinit`**’ senza argomenti si dovrebbe ottenere uno schema simile a quello seguente:

```
...---xinit-+-Xorg
            `-xterm---sh
```

In questo caso si può osservare che ‘**`xinit`**’ avvia il terminale grafico ‘**`xterm`**’, che a sua volta avvia una shell.

28.5.1.2 Utilizzo di «startx»

«

Nella sezione precedente si è visto che è possibile avviare il servente X attraverso ‘**`xinit`**’. Questo modo potrebbe però risultare scomodo quando si ha la necessità di utilizzare sistematicamente determinati attributi. Il sistema grafico dovrebbe essere avviato attraverso lo script ‘**`startx`**’, che è predisposto per ‘**`xinit`**’ nel modo più adatto alle esigenze particolari del proprio sistema.

Di solito le distribuzioni GNU forniscono uno script adattato alla loro impostazione, oppure, lo stesso programma di configurazione di X potrebbe predisporre da solo questo file. In ogni caso, l’amministratore del sistema dovrebbe rivedere questo script ed eventualmente ritoccarlo.

La sintassi di '**startx**', quando si tratta di una versione aderente all'impostazione originale di X, è praticamente uguale a quella di '**xinit**'.

```
startx [ [cliente] opzioni] [ -- [servente] opzioni]
```

Lo script '**startx**' offre però la possibilità di predisporre delle opzioni predefinite per cliente e servente.

```
#!/bin/bash
# Site administrators are STRONGLY urged to write nicer
# versions.
userclientrc=$HOME/.xinitrc
sysclientrc=/etc/X11/xinit/xinitrc
userserverrc=$HOME/.xserverrc
sysserverrc=/etc/X11/xinit/xserverrc
defaultclient=xterm
defaultserver=/usr/bin/X
defaultclientargs=""
defaultserverargs=""
defaultdisplay=":0"
clientargs=""
serverargs=""

enable_xauth=1
...
```

L'esempio mostra come può apparire la parte iniziale di uno script '**startx**'. Sarebbe sufficiente modificare proprio le prime righe per definire delle opzioni predefinite, attribuendo un valore alle variabili '**clientargs**' e '**serverargs**'. La prima si riferisce alle opzioni per il cliente, la seconda per quelle del servente.

Per esempio, volendo avviare il servente, attraverso '**startx**', con una risoluzione di 16 bit/pixel, basterebbe modificare le prime righe come nell'esempio seguente, in modo da fornire al servente l'opzione '**-depth 16**'.

```
userclientrc=$HOME/.xinitrc
sysclientrc=/etc/X11/xinit/xinitrc
userserverrc=$HOME/.xserverrc
sysserverrc=/etc/X11/xinit/xserverrc
defaultclient=xterm
defaultserver=/usr/bin/X
defaultclientargs=""
defaultserverargs=""
defaultdisplay=":0"
clientargs=""
serverargs="-depth 16"
```

Tuttavia, si scorge facilmente la possibilità di usare dei file di configurazione generali per tutto il sistema:

```
sysclientrc=/etc/X11/xinit/xinitrc
sysserverrc=/etc/X11/xinit/xserverrc
```

Pertanto, la possibilità di modificare direttamente lo script è da considerare solo come ultima risorsa.

Se si vuole leggere il contenuto dello script '**start**', ecco in breve la descrizione delle varie fasi in esso contenute.

1. Vengono definite delle variabili per le impostazioni predefinite.
2. Si determina quale script utilizzare per l'avvio dei programmi clienti e quale per l'avvio del servente.
3. Nel ciclo '**while**', vengono scanditi gli eventuali argomenti utilizzati per avviare '**startx**'; se ne vengono trovati, questi

prevalgono su quelli predefiniti.

4. Se ci sono argomenti vengono utilizzati, altrimenti si fa riferimento al contenuto dei file di configurazione.
5. Se non è definita la variabile di ambiente ***XAUTHORITY***, questa viene creata inserendovi il contenuto del file '`~/.Xauthority`'.
6. Definisce l'autorizzazione all'accesso alla stazione grafica (*display*) attraverso una stringa generata in modo casuale, con il programma '**`mcookie`**'.
7. Avvia '**`xinit`**' con gli argomenti determinati in base all'elaborazione precedente.
8. Al termine del funzionamento di '**`xinit`**', elimina l'autorizzazione concessa precedentemente.
9. Infine, viene liberata la memoria usata per l'utilizzo della console virtuale in cui prima si collocava il sistema grafico.

Da quanto visto finora, si può intuire l'importanza dello script '`~/.xinitrc`'. È il mezzo attraverso cui avviare più programmi clienti, ma non solo: esistono programmi che hanno lo scopo di configurare alcune impostazioni del servente X e questo è l'unico posto comodo per metterli in esecuzione in modo automatico. Un esempio di questi programmi è '**`xset`**'.

Supponendo di avere avviato '**`startx`**' senza argomenti, si dovrebbe ottenere uno schema simile a quello seguente:

```
...---startx---xinit-+-Xorg
                     `-fvwm
```

Come si può osservare, rispetto allo stesso esempio visto nella sezione precedente, si ha '**startx**' che avvia '**xinit**', il quale poi provvede al resto.

28.5.1.3 Script «~/.xinitrc»

«

Questo script è quello predefinito per l'avvio dei primi programmi clienti di un servente X avviato attraverso il programma '**xinit**'.

Per preparare il proprio script personalizzato si può partire da quello predefinito della distribuzione GNU che dovrebbe trovarsi all'interno di '`/usr/lib/X11/xinit/`' (oppure '`/etc/X11/xinit/`'). Basta copiarlo nella propria directory personale e cambiargli nome facendolo diventare '`~/.xinitrc`'.

La preparazione di questo script è molto importante, se non altro perché permette di definire il tipo di gestore di finestre che si vuole utilizzare.

Un tempo, il file predefinito era piuttosto complesso, includendo la procedura di autorizzazione all'accesso per la stazione grafica. Recentemente le cose sono cambiate e il problema di questa autorizzazione è stato spostato nello script '**startx**'. Pertanto, se verso la fine del file si incontra un commento del tipo '**# start some nice programs**', si possono aggiungere dei comandi solo dopo quel punto; diversamente, se il file non contiene nulla di particolare, lo si può semplicemente scrivere da zero. L'esempio seguente si riferisce a un'impostazione recente, in cui il file '`~/.xinitrc`' può limitarsi a contenere solo ciò che serve direttamente all'utente finale:

```
#!/bin/sh
xsetroot -solid SteelBlue
exec fvwm
```

Il programma '**xsetroot**' definisce lo sfondo, in questo caso solo un colore, quindi termina immediatamente l'esecuzione. Il programma '**fvwm**' è il gestore di finestre (*window manager*) da avviare; in particolare si usa il comando '**exec**' allo scopo di rimpiazzare la shell. Eventualmente, prima di avviare il gestore di finestre si possono indicare altri programmi che si vuole siano già pronti in esecuzione quando si avvia il servente. Per esempio, volendo avviare '**xclock**' basterebbe modificare le ultime righe come segue:

```
# start some nice programs
xsetroot -solid SteelBlue
xclock -geometry +0+0 &
exec fvwm
```

In questo caso, '**xclock**' viene avviato sullo sfondo perché altrimenti, a differenza di '**xsetroot**', rimarrebbe in funzione fino al ricevimento di un segnale di interruzione, impedendo così l'avvio del gestore di finestre fino al termine del suo funzionamento.[7]

Si deve ricordare che si tratta di uno script, pertanto occorre che gli siano attribuiti i permessi necessari di esecuzione.

28.5.1.4 Configurazione globale e sequenza di script

Quando si vuole fare in modo che si possa mettere in funzione il sistema grafico X senza costringere gli utenti a predisporre la loro personalizzazione tramite il file '`~/.xinitrc`', si deve essere in grado di risalire alla configurazione generale. In questo senso, ogni distribuzione GNU potrebbe avere una propria politica e questo rischia di complicare le cose. Qui viene proposta una situazione, ma in pratica ognuno deve rifare una propria ricerca.

Si parte dallo script '**startx**' per determinare la collocazione dei file di configurazione predefiniti:

```
userclientrc=$HOME/.xinitrc
userserverrc=$HOME/.xserverrc
sysclientrc=/usr/lib/X11/xinit/xinitrc
sysserverrc=/usr/lib/X11/xinit/xserverrc
defaultclient=/usr/bin/X11/xterm
defaultserver=/usr/bin/X11/X
```

In questo caso, si intende intuitivamente che:

- lo script da usare per avviare i programmi clienti, secondo le impostazioni degli utenti, è '`~/.xinitrc`', mentre quello che stabilisce quale sia il programma servente è '`~/.xserverrc`';
- in mancanza degli script degli utenti, si usano '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`' rispettivamente;
- in mancanza anche di questi file, si avvia semplicemente il programma '**xterm**' come cliente e il programma '**X**' come servente.

Tuttavia, dal momento che gli script '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`' servono in pratica alla configurazione del sistema grafico, è normale che la loro collocazione reale sia invece nella directory '`/etc/X11/xinit/`', dove i nomi di origine corrispondono soltanto a dei collegamenti simbolici. Nello stesso modo, il file '`/usr/bin/X11/X`' che rappresenta il servente predefinito, dovrebbe essere un programma che si limita ad avviare a sua volta il file '`/etc/X11/X`', che a sua volta dovrebbe essere un altro collegamento simbolico che punta all'eseguibile corretto (di solito '`/usr/bin/X11/Xorg`').

Giunti a questo punto conviene dare un'occhiata ai file '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`', ovvero a '`/etc/X11/xinit/xinitrc`' e '`/etc/X11/xinit/xserverrc`'. Il file '`xinitrc`' potrebbe presentarsi così:

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

# /etc/X11/xinit/xinitrc
#
# global xinitrc file, used by all X sessions started by
# xinit (startx)

# invoke global X session script
. /etc/X11/Xsession
```

In questo caso, si vede che viene letto il contenuto del file '`/etc/X11/Xsession`' e trattato come una prosecuzione dello script stesso. Attraverso questo script ulteriore, si fanno poi una serie di altre operazioni, con cui si configura in pratica ciò che viene così definito come ***sessione***.

Il sistema grafico X può essere usato senza doversi prendere cura della configurazione della sessione. In pratica, si ottiene questo usando il file '`~/.xinitrc`' personalizzato, perché in tal modo si esclude l'uso dello script '**`xinitrc`**' globale, senza il quale non si attiva lo script '**`Xsession`**'. Tuttavia, se si vogliono usare convenientemente quelli che sono definiti come ***gestori di sessione*** (per esempio Gnome o KDE, che si collocano al di sopra dei comuni gestori di finestre), non si può evitare il passaggio per lo script '**`Xsession`**'.

Senza entrare nel dettaglio dello script '**Xsession**', vale la pena di annotare che questo, se lo trova, utilizza anche il file '`~/.Xsession`', nel caso un utente volesse definire l'utilizzo di un gestore di sessione diverso da quello predefinito.

Volendo dare un'occhiata allo script '**xserverrc**', si potrebbe trovare un contenuto simile a quello seguente:

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

In pratica, si avvia il file '`/usr/bin/X11/X`' ('`/usr/bin/X11/X`'), che, come già descritto, dovrebbe corrispondere in pratica a un collegamento simbolico riferito a '`/etc/X11/X`', il quale, a sua volta, dovrebbe essere un collegamento che punta al servente adatto per il proprio elaboratore.

In questo caso particolare, si vede che, per motivi di sicurezza, sono inibite espressamente le comunicazioni di rete attraverso il protocollo TCP/IP, con l'opzione '**-nolisten tcp**'. Pertanto, un utente che volesse abilitarle, dovrebbe scrivere il proprio file '`~/.xserverrc`', senza l'uso di questa opzione.

28.5.2 Privilegi per il funzionamento di un servente grafico

«

Esiste un particolare importante a proposito del funzionamento di un servente: per poter svolgere il suo compito deve poter accedere a certe risorse disponendo di privilegi adeguati. Perché ciò avvenga e sia consentito l'uso da parte di utenti comuni, è necessario che l'eseguibile che lo rappresenta abbia i permessi necessari a renderlo capace di questo. In pratica deve appartenere all'utente '**root**' e ave-

re il bit SUID attivo (SUID-root). Generalmente, il file '`/usr/bin/X`' è un programma che ottiene tali privilegi e si occupa di avviare il collegamento '`/etc/X11/X`'. L'esempio seguente mostra i permessi di questo file:

```
$ ls -l /usr/bin/X[Invio]

-rwsr-sr-x  1 root  root    7400 gen 29 18:35 /usr/bin/X11/X
```

In questo modo, l'utente comune non può avviare direttamente l'eseguibile del servente grafico che preferisce, ma deve limitarsi a usare '**X**'.

28.5.3 Stazioni grafiche virtuali multiple

X può gestire più di una stazione grafica virtuale simultaneamente, con una modalità d'uso simile a quella delle console virtuali di un sistema GNU/Linux. In pratica, è possibile avviare diversi serventi X a cui si abbina un numero di stazione grafica differente. Dal momento che si tratta sempre della stessa macchina fisica, la configurazione non cambia.

L'avvio di più stazioni grafiche virtuali può creare dei problemi con il mouse se il dispositivo corrispondente non consente la lettura simultanea da parte di più processi. Questo è sempre lo stesso problema legato ai mouse bus e si può risolvere utilizzando il demone '**gpm**' con l'opzione '**-R**', facendo poi in modo che X utilizzi il dispositivo '`/dev/gpmdata`'.

Come è stato descritto nelle sezioni precedenti, il sistema grafico viene avviato generalmente attraverso lo script '**startx**', o even-

tualmente richiamando direttamente il programma '**xinit**'. Quando non si specificano opzioni particolari, si intende voler avviare il servente X utilizzando la stazione grafica '**:0**'. In un sistema GNU/Linux, ciò si traduce in pratica nell'utilizzo della posizione corrispondente alla prima console virtuale disponibile, che di solito è la settima.

Se si vogliono avviare altri serventi X, occorre specificare un diverso numero di stazione grafica, cosa che serve solo a distinguerle. Così, ogni nuovo servente avviato va a utilizzare una posizione corrispondente alla prima console virtuale rimasta libera. In pratica, [*Ctrl Alt F7*] dovrebbe permettere di raggiungere la prima di queste stazioni grafiche virtuali, [*Ctrl Alt F8*] la successiva e così di seguito.

Semplificando quanto mostrato nelle sezioni precedenti, a proposito di '**xinit**' e di '**startx**', si può fare riferimento alla sintassi seguente per avviare un servente X.

```
xinit -- [stazione_grafica] [opzioni]
```

```
startx -- [stazione_grafica] [opzioni]
```

Dopo i due trattini di separazione della parte cliente da quella servente, è possibile indicare il numero della stazione grafica, e subito dopo si possono indicare altre opzioni.

Di solito, si avvia '**startx**' (e meno frequentemente si avvia direttamente '**xinit**') senza indicare alcuna stazione grafica, facendo riferimento implicitamente al numero '**:0**'. Dopo averne avviato uno

con questo numero, non ne possono essere avviati altri con lo stesso, quindi, se si vogliono gestire più serventi contemporaneamente, occorre definire la stazione grafica.

```
$ startx -- :1 [Invio]
```

L'esempio mostrato avvia una copia del servente X utilizzando la stazione grafica '**:1**'.

Ci possono essere dei motivi per avviare diversi serventi X simultaneamente; per esempio per avere due o più sessioni funzionanti in qualità di utenti differenti, oppure per poter confrontare il funzionamento in presenza di diverse opzioni del servente, come nel caso seguente, dove si specifica una profondità di colori di 16 bit.

```
$ startx -- :2 -depth 16 [Invio]
```

È importante tenere a mente che le opzioni del servente, che nell'esempio sono costituite solo da '**-depth 16**', vanno poste dopo l'indicazione della stazione grafica.

28.5.4 Definizione dello schermo

Per l'utilizzo normale che si può fare di X non è necessario doversi rendere conto che ogni programma cliente deve specificare lo schermo su cui vuole apparire. Infatti, viene definita automaticamente la variabile di ambiente ***DISPLAY*** contenente le coordinate dello schermo predefinito. Modificando eventualmente il contenuto di questa variabile, si cambia l'indicazione dello schermo predefinito per i programmi che vengono avviati ricevendo quel valore.

Generalmente è possibile informare un programma dello schermo su cui questo deve apparire attraverso un argomento standard, '**-display**', descritto nella sezione 29.1.

28.5.5 Accedere allo schermo

«

Quando si esegue una sessione TELNET, o qualunque altra cosa che permetta di accedere a un sistema remoto, si avvia una procedura di accesso su un altro elaboratore, utilizzando il proprio come terminale o console remota. Quando si utilizza un servente X è possibile condividere lo schermo del proprio monitor. Per farlo occorre autorizzare l'utilizzo del proprio schermo all'elaboratore remoto. Si osservi il comando seguente:

```
tizio@dinkel.brot.dg:~$ xterm -display :0 &[Invio]
```

Si tratta dell'utente '**tizio**', che dall'elaboratore *dinkel.brot.dg* intende avviare il programma '**xterm**' utilizzando lo schermo '**:0**' presso il suo stesso elaboratore locale. Si osservi anche che se l'utente in questione avvia questo comando da una finestra di terminale che si trova già a funzionare sullo schermo '**:0**', il comando seguente significherebbe la stessa cosa, in quanto l'informazione sullo schermo verrebbe ottenuta dalla variabile di ambiente ***DISPLAY***, senza bisogno di utilizzare l'opzione '**-display**':

```
tizio@dinkel.brot.dg:~$ xterm &[Invio]
```

Questo comando avvia '**xterm**', il quale tenta di connettersi con il servente X che gestisce lo schermo locale '**:0.0**' (abbreviato con '**:0**'), allo scopo di poterlo utilizzare: se il servente X si rifiuta, '**xterm**' deve rinunciare.

L'autorizzazione ad accedere allo schermo deve essere definita anche per lo stesso utente che ha avviato il servente X; tuttavia, questa autorizzazione viene predisposta inizialmente in modo automatico, attraverso '**startx**', oppure uno degli altri script coinvolti.

L'autorizzazione all'utilizzo del proprio schermo grafico da parte di programmi in esecuzione su altri elaboratori connessi in rete può avvenire semplicemente in base a un elenco di indirizzi autorizzati, oppure attraverso altre forme di riconoscimento. Qui vengono spiegati solo i modi più semplici e meno sicuri; per avere una visione completa delle possibilità si devono consultare le pagine di manuale *X(1)*, *xauth(1)* e *Xsecurity(1)*.

È importante non sottovalutare il pericolo di un accesso indesiderato al proprio servente X, in quanto un aggressore preparato può sfruttare questa possibilità per arrivare anche a utilizzare la tastiera. In pratica, un aggressore potrebbe fare tutto quello che gli concedono i privilegi con cui è stato avviato il servente X.

Il metodo più semplice in assoluto per concedere l'accesso al servente X è quello di stabilire attraverso il comando '**xhost**' quali sono gli elaboratori che possono accedere. Questo significa implicitamente che tutti gli utenti di questi elaboratori possono accedere. Volendo distinguere tra gli utenti, occorre utilizzare almeno il metodo delle chiavi in chiaro ('**MIT-MAGIC-COOKIE-1**').

Per attuare in pratica questo secondo meccanismo, viene utilizzato un file di configurazione personale, '`~/.Xauthority`', nel quale

sono elencati degli indirizzi di serventi X e le chiavi di accesso relative. Questo file non è leggibile direttamente; tuttavia, a titolo di esempio, potrebbe contenere le informazioni seguenti, che si riferiscono all'utente '**tizio**' presso il solito elaboratore *dinkel.brot.dg*:

```
dinkel/unix:0  MIT-MAGIC-COOKIE-1  0f207ef0f71e2490b0648c26ed4f3e41
dinkel.brot.dg:0  MIT-MAGIC-COOKIE-1  0f207ef0f71e2490b0648c26ed4f3e41
```

Questo contenuto determina che il servente X, avviato dall'utente a cui appartiene questo file, accetta connessioni locali (attraverso un socket di dominio Unix) e connessioni remote, attraverso la tecnica del '**MIT-MAGIC-COOKIE-1**', quando chi accede fornisce la chiave di riconoscimento '**0f207ef0f71e2490b0648c26ed4f3e41**'. In questo caso, la chiave è la stessa, sia per le connessioni locali, sia per quelle attraverso la rete, ma potrebbero essere diverse; ciò che conta è che il cliente sia in grado di fornire la chiave giusta in base al tipo di connessione che effettua con il servente.

Per fare in modo che il cliente sappia quale chiave utilizzare, occorre che l'utente che tenta di accedere al servente X abbia un file '`~/.Xauthority`' contenente un record adatto. In pratica, se l'utente '**caio**' vuole accedere, deve avere il record seguente nel caso questo avvenga nell'ambito dello stesso elaboratore locale:

```
dinkel/unix:0  MIT-MAGIC-COOKIE-1  0f207ef0f71e2490b0648c26ed4f3e41
```

Oppure, gli serve il record successivo nel caso debba accedere da un altro elaboratore:

```
dinkel.brot.dg:0  MIT-MAGIC-COOKIE-1  0f207ef0f71e2490b0648c26ed4f3e41
```

Lo stesso utente che ha avviato il servente X deve essere autorizzato, attraverso il proprio file '`~/.Xauthority`' che serve per questo scopo, imponendo agli altri la chiave di accesso.

Si può comprendere meglio il meccanismo della chiave di riconoscimento '**`MIT-MAGIC-COOKIE-1`**', solo se si pensa allo scopo che ha: una persona può avere la possibilità di accedere a più elaboratori di una stessa rete locale, ma le utenze relative potrebbero anche corrispondere a nominativi-utente distinti, a seconda dell'elaboratore. Questa persona può avere la necessità di accedere a uno di questi elaboratori, attraverso la rete, avviando lì un programma che però deve apparire presso la stazione da cui sta operando. In altri termini, quando c'è la necessità di avviare un programma che deve apparire sullo schermo di un altro elaboratore, di solito si tratta di utenze che appartengono alla stessa persona fisica; in questo senso non c'è nulla di strano se tutte queste utenze condividono la stessa chiave.

Per la precisione, nel caso di due utenti che appartengono allo stesso elaboratore, il record che descrive la chiave di accesso locale deve essere identico per entrambi. Di conseguenza, la condivisione di questo implica che il servente X avviato da uno di questi due è anche accessibile dall'altro.

Dal momento che il file '`~/.Xauthority`' non è un file di testo normale, per accedervi, si utilizza generalmente il programma '**`xauth`**'.

28.5.5.1 Utilizzo di «xauth»

«

Il programma '**xauth**' è necessario per poter accedere alle informazioni contenute nei file di autorizzazione, normalmente '`~/.Xauthority`', per poterle modificare. Per la maggior parte delle situazioni, '**xauth**' non ha bisogno di contattare il servente X.

```
xauth [opzioni] [comando argomento...]
```

Il programma '**xauth**' interviene in base a dei comandi, che gli possono essere impartiti come argomenti della stessa riga di comando, nella parte finale, oppure in modo interattivo, attraverso l'invito seguente:

```
xauth>
```

Spesso, i comandi richiedono l'indicazione di un file. In quella occasione, se si utilizza un trattino singolo ('**-**'), questo viene inteso come lo standard input, oppure lo standard output, a seconda del contesto.

Tabella 28.26. Alcune opzioni.

Opzione	Descrizione
`-f` ***file_di_autorizzazione***	Permette di accedere a un file di autorizzazioni differente da quello standard, che di solito è '`~/.Xauthority`'.

Opzione	Descrizione
`-b`	L'accesso al file delle autorizzazioni è regolato attraverso un file lucchetto (*lock file*), che alle volte potrebbe rimanere presente senza che ce ne sia più bisogno. Eccezionalmente e con prudenza, si può utilizzare questa opzione per forzare il blocco ed eliminare il file lucchetto (*lock file*) relativo.

Tabella 28.27. Alcuni comandi. I comandi di '**`xauth`**' possono essere impartiti in modo interattivo, oppure possono essere indicati come argomenti finali della riga di comando di '**`xauth`**'.

Comando	Descrizione
`add` ***stazione_grafica*** ↩ ↪ ***protocollo*** ↩ ↪ ***chiave_esadecimale***	Questo comando serve ad aggiungere manualmente un record nel file di autorizzazione. Deve essere specificata: la stazione grafica, ovvero un indirizzo che non arriva a specificare anche lo schermo (in caso contrario questa informazione viene ignorata semplicemente); il tipo di protocollo, che può anche essere abbreviato con un punto singolo ('`.`'), nel caso si tratti del tipo '**`MIT-MAGIC-COOKIE-1`**'; la chiave esadecimale, ovvero una stringa composta da un numero pari di cifre esadecimali, senza alcun prefisso.

Comando	Descrizione
`list` [*stazione_grafica*...]	Permette di visualizzare i record del file di autorizzazione, limitandosi alle stazioni grafiche indicate. Se queste non sono specificate, il comando mostra l'elenco completo.
`info`	Permette di conoscere alcune informazioni generali sul file di autorizzazione.
`extract` *file* [*stazione_grafica*...] `nextract` *file* ↩ ↪ *stazione_grafica*...	Questo comando permette di estrarre alcuni record dal file delle autorizzazioni, corrispondenti alle stazioni grafiche indicate. Il risultato viene accumulato nel file indicato come primo argomento di questo comando. Nel primo caso, con '**`extract`**', le informazioni vengono memorizzate in forma binaria, mentre nel secondo, con '**`nextract`**', queste informazioni sono convertite in forma testuale.

Comando	Descrizione
`merge` ***file*** `nmerge` ***file***	Questo comando consente di acquisire nel file di autorizzazione i record contenuti nel file indicato. Questi record vanno a sostituire quelli corrispondenti, riferiti alle stesse stazioni grafiche che dovessero essere già presenti nel proprio file di autorizzazione. Anche in questo caso vale la differenza per cui '**merge**' si aspetta di attingere i record da un file binario, mentre '**nmerge**' utilizza un file di testo normale.
`remove` ***stazione_grafica*** …	Elimina i record specificati attraverso l'indicazione delle stazioni grafiche relative.

`exit`
 	
`quit`
	Questi due comandi riguardano il funzionamento interattivo di '**xauth**'. Con '**exit**' viene concluso il funzionamento del programma, salvando le modifiche; con '**quit**', si ottiene una conclusione senza salvare.

Segue la descrizione di alcuni esempi.

- `tizio@dinkel.brot.dg:~$` **`xauth add :0 . 12345678`** [*Invio*]

  L'utente aggiunge, o modifica, il record di autorizzazione riferito all'accesso locale, specificando per questo il protocollo '**MIT-MAGIC-COOKIE-1**' in modo predefinito, attraverso

il punto, indicando una stringa esadecimale molto semplice: 12345678_{16}.

- `tizio@dinkel.brot.dg:~$` **`extract /tmp/prova :0`** [*Invio*]

  Estrae una copia del record di autorizzazione all'accesso locale e la salva nel file '`/tmp/prova`'.

- `caio@dinkel.brot.dg:~$` **`merge /tmp/prova :0`** [*Invio*]

  Un altro utente, si appropria dei record contenuti nel file '`/etc/prova`'.

- `tizio@roggen.brot.dg:~$` **`xauth extract - $DISPLAY;`** ↩
  ↪ **`| rsh dinkel.brot.dg xauth`** ↩
  ↪ **`merge -`** [*Invio*]

  L'utente '**`tizio`**' che sta utilizzando l'elaboratore *roggen.brot.dg* ottiene attraverso '**`rsh`**' di aggiungere al proprio file di autorizzazione remoto, quello presso la sua utenza corrispondente nell'elaboratore *dinkel.brot.dg*, il record riferito al servente X che sta utilizzando in quel momento. In altri termini, fa in modo di poter avviare dei programmi presso l'elaboratore remoto, utilizzando la stazione grafica su cui si trova. Si osservi l'uso della variabile di ambiente ***DISPLAY*** per ottenere l'indicazione precisa dello schermo che sta utilizzando e anche l'uso del trattino per collegare i due programmi attraverso i flussi standard.

28.5.5.2 Utilizzo di «mcookie»

«

Il programma '**`mcookie`**' ha lo scopo di generare un numero esadecimale, più o meno casuale, convertito in stringa, che viene emesso attraverso lo standard output:

```
mcookie
```

La sua utilità sta solo nel facilitare la generazione di chiavi per il sistema di autorizzazione. La situazione più comune in cui viene utilizzato è il comando seguente, dove in pratica ci si risparmia di decidere la chiave:

```
$ xauth add :0 . `mcookie` [Invio]
```

28.5.5.3 Riepilogo sull'utilizzo del file di autorizzazione

Il file di autorizzazione è composto da record contenenti tre informazioni: la stazione grafica (senza il dettaglio dello schermo); il nome di un protocollo di autenticazione; una chiave, il cui significato varia a seconda del tipo di protocollo utilizzato.

È importante sottolineare che può esistere un solo record per stazione grafica, per cui, ogni volta che si aggiunge un record per una certa stazione, questo va a sostituire un altro record eventuale riferito alla stessa stazione.

In generale, si distingue tra la stazione grafica locale, a cui si accede senza passare per la rete, e le stazioni grafiche remote, che contengono anche l'indicazione del nome del nodo di rete. Tra le stazioni remote ci può essere anche quella locale, indicata secondo il punto di vista della rete.

Perché possa avvenire una connessione tra un programma cliente e un servente X, è necessario che il record di autorizzazione a cui può accedere il cliente, riferito al servente X in questione, sia identico a quello corrispondente del servente X.

Il sistema di autorizzazione di X sembra fatto perché le chiavi siano cambiate spesso. In generale, si cerca di sistemare l'autorizzazione sempre solo nel momento in cui ne esiste il bisogno, ma subito dopo sarebbe bene cambiare la chiave di autorizzazione.

28.5.5.4 Utilizzo di «xhost»

Il programma '**xhost**' permette di aggiungere o togliere nomi dalla lista di elaboratori e utenti a cui è concesso di utilizzare lo schermo grafico, senza la richiesta di altre forme di autenticazione:

```
xhost [[+|-]nome...]
```

```
xhost [+|-]
```

Se non vengono utilizzati argomenti, '**xhost**' emette un messaggio informando sullo stato attuale del controllo degli accessi. I nomi indicati nella sintassi di '**xhost**' hanno una struttura particolare:

```
famiglia:indirizzo
```

In pratica, per le connessioni su reti IPv4 si utilizza la famiglia '**inet**'.

Le funzionalità di X non sono sempre presenti su tutte le piattaforme. In questo caso particolare, potrebbe darsi che non sia possibile regolare gli accessi ai singoli utenti.

Se si vuole concedere sistematicamente l'accesso a qualche nodo di rete, conviene inserire i comandi necessari all'interno del file

‘`~/.xinitrc`’ in modo che siano eseguiti ogni volta all’avvio del servente X.

Tabella 28.28. Alcune opzioni.

Opzione	Descrizione
`+`	L’accesso è consentito a tutti.
`-`	L’accesso è consentito solo agli elaboratori e agli utenti inclusi nell’elenco di quelli autorizzati.
[`+`] ***nome***	Il nome indicato -- può trattarsi di un elaboratore o di un utente di un elaboratore -- è autorizzato a utilizzare lo schermo. Il segno ‘**+**’ iniziale è facoltativo.
`-`***nome***	Il nome indicato (può trattarsi di un elaboratore o di un utente di un elaboratore) non è autorizzato a utilizzare lo schermo. Le connessioni in corso non vengono interrotte, ma le nuove connessioni vengono impedite.

Segue la descrizione di alcuni esempi.

- `$` **`xhost +`**[*Invio*]

  Autorizza chiunque ad accedere.

- `$` **`xhost -`**[*Invio*]

  Limita la possibilità di accesso ai soli nomi inseriti nell’elenco di elaboratori e utenti autorizzati.

- `$` **`xhost +inet:roggen.brot.dg`**[*Invio*]

Consente all'elaboratore *roggen.brot.dg* di accedere al servente grafico.

- `$` **`xhost -inet:roggen.brot.dg`** [*Invio*]

  Elimina l'elaboratore *roggen.brot.dg* dalla lista di quelli a cui è consentito accedere.

28.5.5.5 Utilizzo di «xon»

«

'**`xon`**' esegue un comando in un elaboratore remoto attraverso '**`rsh`**', facendo in modo che venga utilizzato il servente X locale:

```
xon nodo_remoto [opzioni] [comando]
```

Si tratta in pratica di un modo abbreviato per eseguire un'applicazione remota senza la necessità di utilizzare la solita opzione '**`-display`**'.[8]

Se attraverso gli attributi non viene indicato alcun comando da eseguire, '**`xon`**' tenta di avviare '**`xterm -ls`**', in pratica una sessione '**`xterm`**' di *login*.

'**`xon`**' è in grado di funzionare solo quando l'elaboratore remoto è configurato in modo da consentire le connessioni remote attraverso '**`rsh`**' senza richiedere alcun tipo di riconoscimento. Sotto questo aspetto, '**`xon`**' è limitato all'utilizzo nelle reti chiuse in cui esiste un serio rapporto di fiducia tra le persone che vi accedono.

Tabella 28.29. Alcune opzioni.

Opzione	Descrizione
`-access`	Prima di eseguire il comando indicato, utilizza '**xhost**' nel tentativo di autorizzare l'elaboratore remoto a utilizzare il proprio servente X. In effetti, lo scopo di '**xon**' è quello di facilitare l'esecuzione di programmi remoti ma con un I/O locale, cioè attraverso il servente X con il quale si interagisce.
`-debug`	Quando '**xon**' viene avviato attraverso una finestra di terminale, utilizzando questa opzione si riceve lo standard output e lo standard error. In tal modo si possono conoscere eventuali segnalazioni di errore e qualunque altro output normale.

L'esempio seguente mostra l'avvio del programma '**xcalc**' nell'elaboratore *roggen.brot.dg*, utilizzando il servente X locale. Prima di farlo, '**xon**' avvia '**xhost**' per consentire all'elaboratore remoto di accedere al proprio servente X.

```
$ xon roggen.brot.dg -access /usr/bin/X11/xcalc [Invio]
```

28.5.6 Accedere allo schermo con Secure Shell

Secure Shell (sezione 44.7) facilita le connessioni remote, gestendo in modo automatico tutto il procedimento di autorizzazione all'accesso al proprio schermo. Per arrivare a questo risultato, è comunque necessario abilitare tale funzionalità nella configurazione: sia dalla parte del servente, sia dalla parte del cliente.

Nel file di configurazione del servente Secure Shell, è necessario trovare queste direttive:

```
X11Forwarding yes
X11DisplayOffset 10
```

Nel file di configurazione del cliente Secure Shell, è necessario trovare queste direttive:

```
Host *
    ForwardX11 yes
```

Così facendo, una volta aperta una finestra di terminale, ci si può collegare all'elaboratore remoto usando il cliente Secure Shell, come nell'esempio seguente:

```
tizio$dinkel.brot.dg:~$ ssh caio@roggen.brot.dg[Invio]
```

Dopo la fase di autenticazione, che potrebbe consistere nella richiesta della parola d'ordine, è possibile verificare che la variabile di ambiente ***DISPLAY*** risulta impostata in modo da fare riferimento al proprio elaboratore locale, utilizzando uno schermo particolare, come definito nella direttiva '**X11DisplayOffset**':

```
caio$roggen.brot.dg:~$ echo $DISPLAY[Invio]

  dinkel.brot.dg:10.0
```

A questo punto, è sufficiente avviare un programma grafico qualunque nell'elaboratore remoto, senza bisogno di altro: si ottiene di farlo funzionare sul proprio schermo grafico.

```
caio$roggen.brot.dg:~$ xclock[Invio]
```

Si osservi che la comunicazione tra i due elaboratori avviene all'interno di un tunnel definito da Secure Shell. Ciò consente di ottenere una connessione cifrata; in ogni caso, tuttavia, è da tenere in considerazione che non viene rilevata come tale da un programma di analisi del traffico in rete, ma solo come una connessione di Secure Shell.

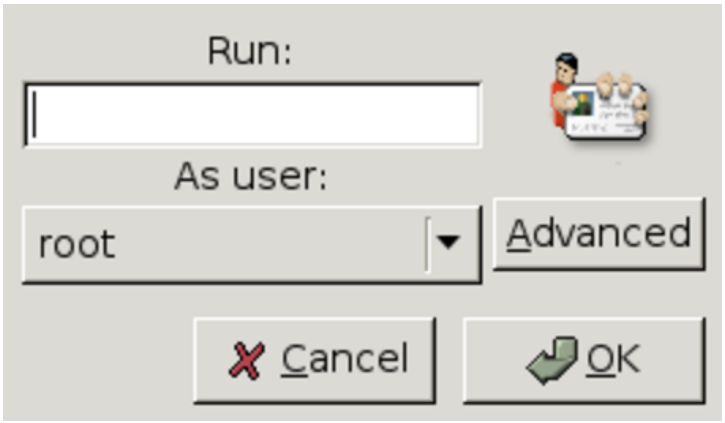
28.5.7 Cambiare identità

Quando si intende avviare un programma che utilizza la grafica, ma con i privilegi di un altro utente, non basta usare il comando '**su**', come avviene per quei programmi che richiedono soltanto un terminale a caratteri. Infatti, si crea il problema delle autorizzazioni, già descritto nelle sezioni precedenti. Pertanto, occorre eventualmente un programma analogo a '**su**', in grado però di provvedere anche a ciò che serve per autorizzare la comunicazione con il proprio servente X. A titolo di esempio viene descritto Gksu:

```
gksu [-u utente] [opzioni] [comando]
```

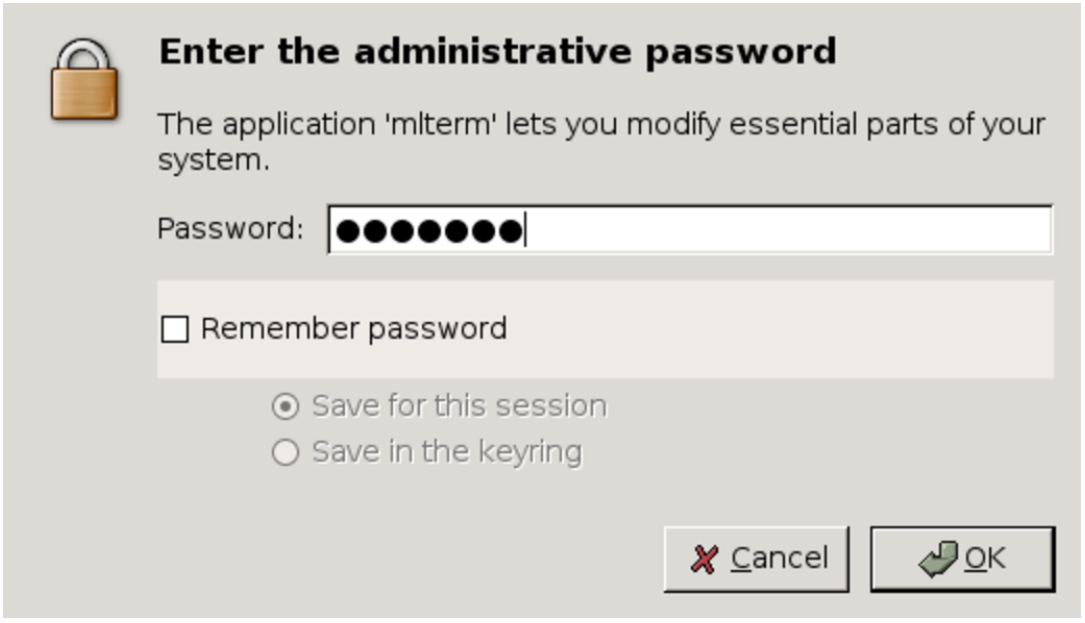
Se il programma '**gksu**' viene avviato senza alcun argomento, chiede interattivamente cosa deve fare: quale utente deve impersonare e quale comando eseguire.

Figura 28.33. Il programma ‘**gksu**’ avviato senza alcun argomento chiede tutto in modo interattivo.



In ogni caso, viene richiesto l’inserimento della parola d’ordine dell’utente che interessa. Di norma, la richiesta della parola d’ordine coincide con un congelamento momentaneo delle altre attività.

Figura 28.34. La richiesta dell’inserimento della parola d’ordine. In questo caso verrebbe avviato il programma ‘**mlterm**’ in qualità di utente ‘**root**’.



28.5.8 Tipi di carattere

In base a quanto indicato nel file di configurazione ‘`/etc/X11/xorg.conf`’ nella sezione ‘**`Files`**’, i tipi di carattere utilizzati da X sono collocati nelle directory successive a ‘`/usr/lib/X11/fonts/`’. All’interno di queste directory si trovano dei file contenenti le informazioni sui vari tipi di carattere tipografico e i loro nomi sono contenuti negli elenchi ‘`fonts.dir`’.

Il nome di un carattere tipografico è strutturato in un modo altamente descrittivo. Segue un esempio che viene scomposto.[9]

```
-b&h-lucidatypewriter-medium-r-normal-sans-8-80-100-100-m-50-iso8859-1
```

- ‘**`b&h`**’ è la «fonderia», ovvero il produttore;
- ‘**`lucidatypewriter`**’ definisce la famiglia;
- ‘**`medium`**’ è lo spessore;
- ‘**`r`**’ è il tipo -- «r» sta per *roman* (tondo), «i» indicherebbe *italic* (corsivo) e «o» *oblique* (obliquo);
- ‘**`normal`**’ è l’ampiezza orizzontale;
- ‘**`sans`**’ è una particolarità dello stile, in questo caso indica l’assenza di grazie o *serif*, cioè dei terminali che facilitano la lettura;
- ‘**`8`**’ indica la dimensione in punti grafici (pixel);
- ‘**`80`**’ indica la dimensione in decimi di punto tipografici (precisamente si tratta di 722,7 per pollice, pari a circa 0,035 mm);
- ‘**`100`**’ è la risoluzione orizzontale, espressa in punti per pollice, per la quale è stato progettato il tipo di carattere;

- '**100**' è la risoluzione verticale, espressa in punti per pollice, per la quale è stato progettato il tipo di carattere;
- '**m**' è la larghezza, in questo caso *monospaced*, ovvero a larghezza fissa, mentre l'alternativa sarebbe «p», cioè *proportional*;
- '**50**' è lo spessore medio espresso in decimi di punto (in questo caso si tratta di cinque punti normali);
- '**iso8859-1**' è l'insieme di caratteri, in questo caso, il codice '**iso8859-1**' corrisponde al noto ISO Latin 1.

28.5.9 X e l'uso senza dispositivo di puntamento

«

L'utilizzo del sistema grafico senza mouse, o senza un dispositivo equivalente, può essere importante in condizioni di emergenza, o comunque quando il tipo di mouse che si ha a disposizione potrebbe risultare più scomodo che altro.

I serventi grafici di X offrono queste funzionalità attraverso il tastierino numerico, dopo aver attivato le estensioni della tastiera. Perché ciò sia possibile è necessario che nel file di configurazione sia commentata l'istruzione che si in questo esempio, oppure che sia assente del tutto:

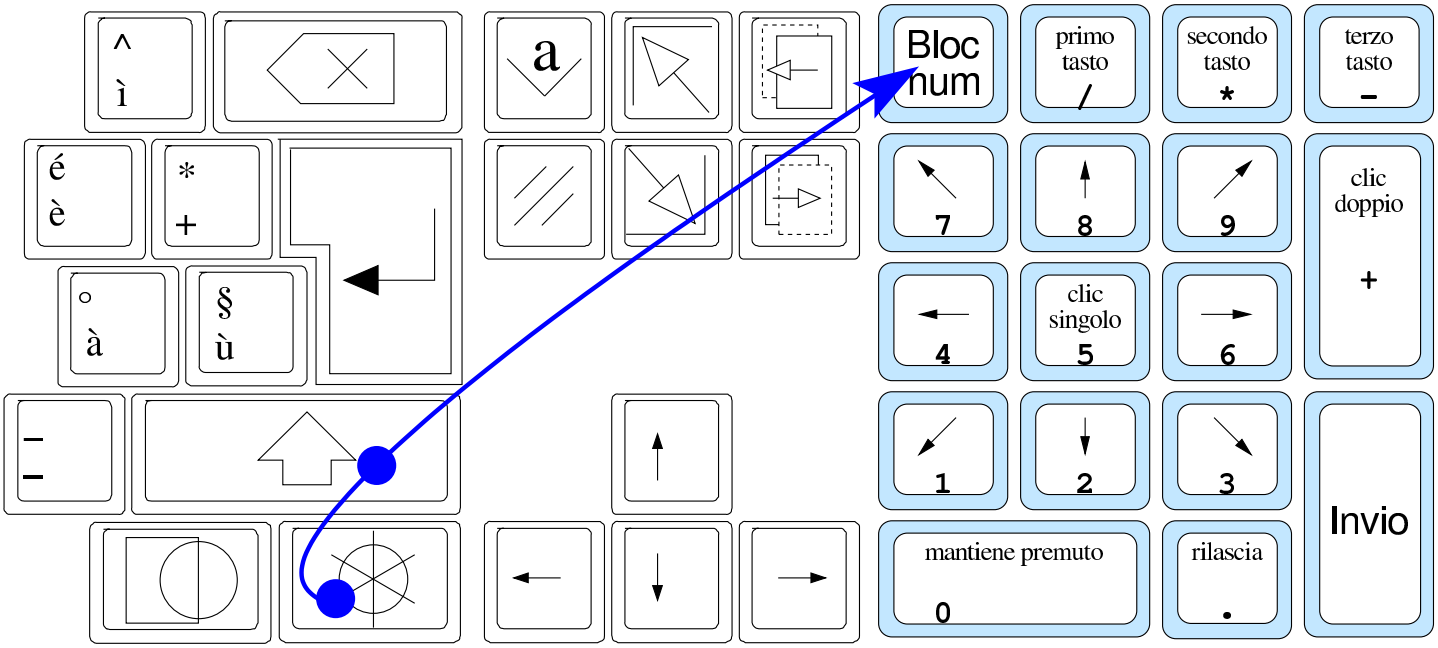
```
# Option    XkbDisable
```

Per abilitare l'uso del tastierino numerico in modo che possa sostituirsi al mouse occorre utilizzare la combinazione [*Ctrl Maiuscole BlocNum*] («control», «maiuscole», «blocconumeri»). Se la combinazione riesce si ottiene una segnalazione sonora (se si ripete la combinazione si disabilita l'uso del tastierino).

Da quel momento, si possono utilizzare i tasti [4], [8], [6] e [2], per spostare il puntatore rispettivamente verso sinistra, in alto, a destra e in basso. Inoltre, si possono usare anche i tasti [7], [9], [3] e [1], per ottenere degli spostamenti obliqui. Questi spostamenti potrebbero essere piuttosto lenti in condizioni normali; per accelerarli, mentre si tiene premuto il tasto che si riferisce alla direzione scelta, si può premere e rilasciare immediatamente un altro tasto, scegliendolo in modo tale che in quel momento non abbia un significato particolare; probabilmente la cosa migliore è usare per questo il tasto delle maiuscole.

Per emulare i tasti del mouse si utilizzano gli altri tasti del tastierino numerico: [5] corrisponde a un clic; [+] corrisponde a un clic doppio; [0] rappresenta la pressione di un tasto senza rilasciarlo; [.] rilascia il tasto del mouse. In condizioni normali, ciò corrisponde al primo tasto del mouse, ma si può specificare precisamente il tasto attraverso una combinazione con i tasti [/], [*] e [-], che rappresentano rispettivamente il primo, il secondo (quello centrale) e il terzo tasto del mouse. Per esempio, [* 5] corrisponde a un clic con il tasto centrale del mouse.

Figura 28.37. Comandi per l'emulazione del mouse con X.



X, dopo un po' di tempo in cui non si utilizza più il tastierino numerico in sostituzione del mouse, ne disabilita l'emulazione in modo automatico.

28.6 La tastiera e la sua configurazione «elementare»

«

La configurazione della tastiera con X è un problema molto delicato, che normalmente si risolve semplicemente utilizzando le opzioni che sono già state stabilite; tuttavia, manca una documentazione adeguata e coerente per le opzioni esistenti e ancora di più manca una documentazione sufficiente per poter gestire in proprio una mappa autonoma della tastiera.

Si tenga in considerazione che, a causa della complessità del sistema di gestione della tastiera di XFree86 e di X.Org, le indicazioni

che appaiono in questo capitolo possono essere imprecise o troppo semplificate.

28.6.1 Livelli, tasti morti, composizione e gruppi

La tastiera dell'elaboratore deriva concettualmente da quella delle prime macchine da scrivere, con le quali ogni tasto poteva generare due simboli diversi, alzando o abbassando il blocco dei martelli. Per questa ragione, nelle tastiere per elaboratore di lingua inglese i tasti delle maiuscole si chiamano *shift*, perché spostano virtualmente questi martelli. Secondo lo standard ISO 9995, la selezione di un simbolo in base al meccanismo che normalmente è controllato attraverso il tasto delle maiuscole, avviene in base a dei ***livelli***. In pratica, su uno stesso tasto ci può essere una lettera minuscola o maiuscola, o comunque due simboli differenti, in base alla selezione del livello appropriato.

Lo standard ISO 9995 considera che i livelli possano essere anche più di due; per esempio su diverse tastiere europee si ottiene la selezione di un terzo livello usando una combinazione con il tasto [*AltGr*]. Con XFree86 e X.Org si gestiscono normalmente quattro livelli, ma bisogna **tenere premuto inizialmente il tasto [*AltGr*] e aggiungere eventualmente il tasto [*Maiuscole*] subito dopo**, per raggiungere il quarto livello.

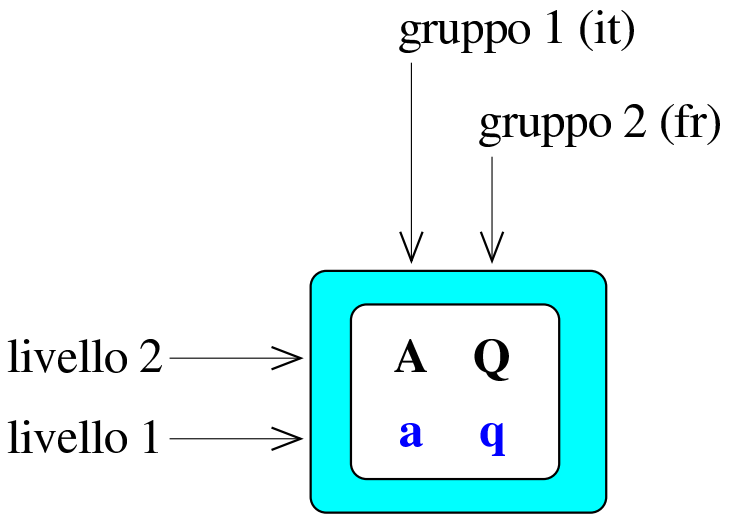
Generalmente, con l'uso dei vari livelli disponibili dovrebbe essere possibile esaurire l'insieme dei simboli necessari a una certa lingua; tuttavia spesso si utilizzano altre due tecniche per facilitare l'inserimento di caratteri particolari: tasti morti che servono ad accentare

la lettera successiva e sequenze di composizione. Nel primo caso, la pressione di un tasto (direttamente o attraverso la selezione del livello appropriato) non genera alcun carattere, in attesa del tasto successivo: se l'associazione tra i due è valida si ottiene la lettera accentata corrispondente. Nel secondo caso viene utilizzato un tasto per richiedere espressamente una modalità di composizione, a cui segue l'inserimento di una sequenza di simboli appropriati: se la sequenza viene riconosciuta, si ottiene un carattere «composto».

Nelle mappe europee comuni, XFree86 e X.Org offrono la funzione di composizione nel livello due del tasto [*AltGr*], ovvero con la combinazione [*Maiuscole AltGr*]. Quello che si deve osservare è che la combinazione deve iniziare con la pressione di [*Maiuscole*], perché facendo il contrario, si selezionerebbe invece il quarto livello.

Per poter selezionare un insieme di simboli differente, si possono definire dei *gruppi* alternativi, da attivare a scelta. Per esempio, associando a un primo gruppo la mappa della tastiera italiana e a un secondo la mappa francese, in corrispondenza di un tasto si potrebbe ottenere sia la lettera «a» minuscola e maiuscola (tastiera italiana), sia la lettera «q» minuscola e maiuscola (tastiera francese).

Figura 28.38. Esempio di un tasto con due gruppi, ognuno con due livelli: la lettera «a» appartiene al primo livello del primo gruppo; la lettera «A» appartiene al secondo livello del primo gruppo; la lettera «q» appartiene al primo livello del secondo gruppo; la lettera «Q» appartiene al secondo livello del secondo gruppo.

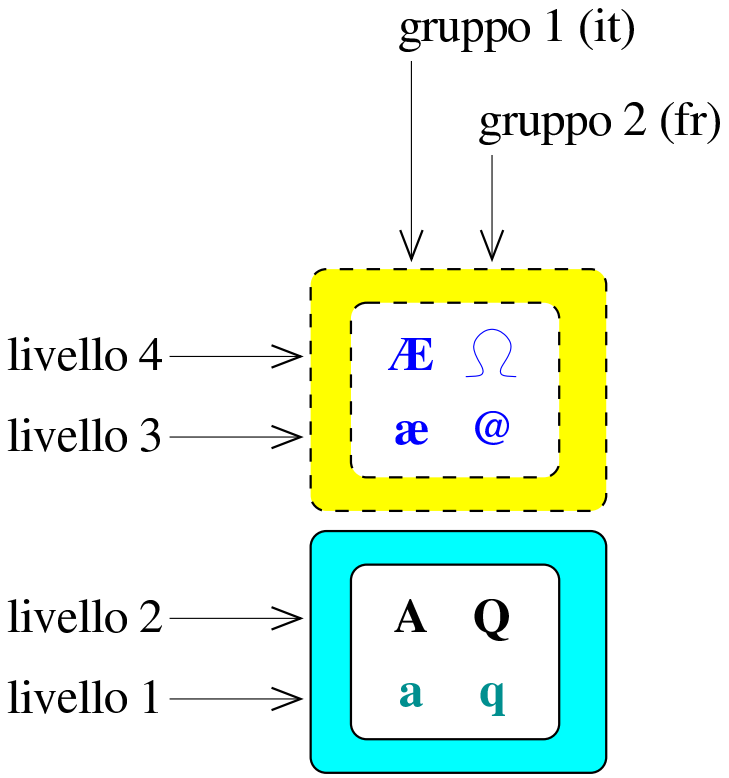


XFree86 e X.Org gestiscono facilmente fino a quattro gruppi, ma quello che conta è avere il mezzo per selezionarli.

Come è noto, quando i due livelli servono a selezionare la stessa lettera, ma in forma minuscola o maiuscola, di norma sul tasto fisico della tastiera appare solo il simbolo corrispondente alla lettera maiuscola.

Come si intende facilmente, per ottenere il passaggio a un livello o a un gruppo differente da quello attivo, si devono usare tasti o combinazioni di tasti appropriati. I tasti usati per questo scopo vengono chiamati normalmente ***modificatori***.

Figura 28.39. Normalmente XFree86 e X.Org gestiscono quattro livelli. Quello che si vede in questa figura è il confronto tra il gruppo '**it**' e '**fr**' per lo stesso tasto.



Riquadro 28.40. Confusione tra livelli e gruppi.

La complessità della configurazione della tastiera con XFree86 e X.Org è tale per cui tra gruppi e livelli si crea un po' di confusione. Infatti, di solito la definizione dei simboli associati ai livelli superiori al secondo avviene incorporando quelli di un altro gruppo. Pertanto, i gruppi vanno considerati in base al contesto e non in senso «assoluto», tenendo conto che il cambiamento di gruppo può voler dire che si sta passando al terzo livello, oppure che si sta scambiando la mappa attuale con quella di un'altra lingua.

È bene osservare che, a seconda di come è configurata la tastiera, il tasto [*Fissamaiuscole*] può avere funzionalità diverse, anche se intuitivamente ciò non si dovrebbe apprezzare. In pratica, in alcune

tastiere europee, come nel caso di quella francese, la pressione di questo tasto attiva e mantiene il secondo livello, come se fosse sempre premuto il tasto delle maiuscole normale; in altri casi, come avviene nella tastiera statunitense e derivate (compresa quella italiana), il comportamento è diverso. Infatti, in questi casi ciò che si vuole è di ottenere le lettere maiuscole solo nella porzione alfanumerica della tastiera, mentre il resto dei simboli deve essere accessibile nella stessa modalità di prima; inoltre, se si preme il tasto delle maiuscole, si vuole che il risultato che si ottiene quando il tasto [*Fissamaiuscole*] è inserito sia di nuovo di lettere minuscole. Per questo tipo di trasformazioni, pertanto, il tasto [*Fissamaiuscole*] inserito non serve a richiedere il cambiamento di livello, ma **la trasformazione della lettera** da minuscola a maiuscola, o viceversa.

Questa caratteristica è molto importante e permette di capire il motivo, per cui, nella configurazione della tastiera italiana, si ottengono le lettere accentate maiuscole inserendo proprio il [*Fissamaiuscole*].

28.6.2 Configurazione elementare

La configurazione della mappa della tastiera si fa normalmente all'interno del file di configurazione di X e si può alterare durante il funzionamento con l'aiuto del programma '**`setxkbmap`**'.[10] Nella situazione più semplice si fa riferimento a un modello di tastiera attraverso le indicazioni contenute nei file della directory '`/etc/X11/xkb/rules/`'. Per esempio, si potrebbero leggere le direttive seguenti nel file di configurazione di X:

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
        Option      "CoreKeyboard"
        Option      "AutoRepeat"    "500 30"
        Option      "LeftAlt"       "Meta"
        Option      "RightAlt"      "ModeShift"
        Option      "XkbRules"      "xorg"
        Option      "XkbModel"      "pc105"
        Option      "XkbLayout"     "it"
        Option      "XkbVariant"    "basic"
        Option      "XkbOptions"
                    "terminate:ctrl_alt_bksp"
EndSection
```

Le direttive significative dell'esempio sono quelle riferite alle opzioni che iniziano con la sigla '**`Xkb`**'. L'opzione '**`XkbRules`**' dichiara delle «regole» generali; nella maggior parte dei casi va bene indicare la sigla '**`xorg`**', che richiama l'utilizzo dei file '`xkb/rules/xorg`' e '`xkb/rules/xorg.lst`'.

Seguendo l'esempio, all'interno del file '`xkb/rules/xorg.lst`' si possono individuare tutte le voci che possono essere utilizzate per l'opzione '**`XkbModel`**':

```
! model
  pc101          Generic 101-key PC
  pc102          Generic 102-key (Intl) PC
  pc104          Generic 104-key PC
  pc105          Generic 105-key (Intl) PC
…
…
  trust          Trust Wireless Keyboard Classic
  trustda        Trust Direct Access Keyboard
  yahoo          Yahoo! Internet Keyboard
```

Nell'esempio si vede la selezione di una tastiera a 105 tasti comune; proseguendo, si vede l'opzione '**XkbLayout**' a cui viene attribuita la voce '**it**', che presumibilmente serve a individuare una mappa adatta alla lingua italiana. Anche la disponibilità di queste sigle dipende dal contenuto del file '`xkb/rules/xorg.lst`', che a questo proposito si può mostrare così:

```
! layout
  us             U.S. English
  en_US          U.S. English w/ ISO9995-3
  us_intl        U.S. English w/ deadkeys
  al             Albanian
…
…
  it             Italian
  jp             Japanese
  kan            Kannada
…
…
  yu             Yugoslavian
  nec/jp         PC-98xx Series
```

La disposizione dei tasti associata alla sigla assegnata all'opzione

‘**XkbLayout**’ può prevedere delle varianti e se necessario quella preferita va specificata con l’opzione ‘**XkbVariant**’. Nell’esempio la dichiarazione riferita all’opzione ‘**XkbVariant**’ è perfettamente inutile, dal momento che con la parola chiave ‘**basic**’ non si specifica alcunché. Eventualmente, nel file ‘`xkb/rules/xorg.lst`’ si può vedere quali varianti sono disponibili, ammesso che funzionino:

```
! variant
  nodeadkeys    Eliminate dead keys
```

Volendo gestire più gruppi di simboli, si possono indicare più alternative tra le opzioni di ‘**XkbLayout**’, come nell’esempio seguente:

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
        Option      "CoreKeyboard"
        Option      "AutoRepeat"    "500 30"
        Option      "LeftAlt"       "Meta"
        Option      "RightAlt"      "ModeShift"
        Option      "XkbRules"      "xorg"
        Option      "XkbModel"      "pc105"
        Option      "XkbLayout"     "it,fr,de"
        Option      "XkbVariant"    "basic,basic,basic"
        Option      "XkbOptions"
                    "grp:alt_shift_toggle,terminate:ctrl_alt_bksp"
EndSection
```

In questa circostanza, si vogliono gestire tre mappe alternative: italiana, francese e tedesca (se ne possono gestire al massimo quattro). Per passare da un gruppo al successivo si deve usare la combinazione di tasti [*Alt Maiuscole*] (è indifferente se si tratta di quelli a sinistra o a destra), come si intuisce dall’opzione ‘**XkbOptions**’. Anche per

sapere cosa si può usare per selezionare un gruppo si deve consultare il file ‘`xkb/rules/xorg.lst`’:

```
! option
  grp                    Group Shift/Lock behavior
  grp:switch             R-Alt switches group while pressed
  grp:lwin_switch        Left Win-key switches group while pressed
  grp:rwin_switch        Right Win-key switches group while pressed
  grp:win_switch         Both Win-keys switch group while pressed
  grp:toggle             Right Alt key changes group
  …
  …
  grp:ctrl_alt_toggle    Alt+Control changes group
  grp:alt_shift_toggle   Alt+Shift changes group
  grp:menu_toggle        Menu key changes group
  …
```

Con l’aggiunta di diverse voci nell’opzione ‘**`XkbLayout`**’, è stata ampliata nello stesso modo la voce relativa alle varianti. Nel caso una delle disposizioni richiedesse una variante particolare, si potrebbe inserire rispettando l’ordine:

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
        Option      "CoreKeyboard"
        Option      "AutoRepeat"    "500 30"
        Option      "LeftAlt"       "Meta"
        Option      "RightAlt"      "ModeShift"
        Option      "XkbRules"      "xorg"
        Option      "XkbModel"      "pc105"
        Option      "XkbLayout"     "it,fr,de"
        Option      "XkbVariant"    "basic,nodeadkeys,basic"
        Option      "XkbOptions"
                    "grp:alt_shift_toggle,terminate:ctrl_alt_bksp"
EndSection
```

In questo caso si specifica la variante '**nodeadkeys**' per la disposizione francese, che in pratica annulla l'uso dei tasti morti in quel contesto.

In caso di difficoltà, l'opzione '**XkbOptions**' può servire per specificare cosa usare per richiamare il terzo livello:

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
        Option      "CoreKeyboard"
        Option      "AutoRepeat"    "500 30"
        Option      "LeftAlt"       "Meta"
        Option      "RightAlt"      "ModeShift"
        Option      "XkbRules"      "xorg"
        Option      "XkbModel"      "pc105"
        Option      "XkbLayout"     "it,fr,de"
        Option      "XkbVariant"    "basic,nodeadkeys,basic"
        Option      "XkbOptions"
             "grp:alt_shift_toggle,lv3:switch,terminate:ctrl_alt_bksp"
EndSection
```

In questo caso si dichiara che si vuole usare il tasto [*Ctrl*] alla destra per passare al terzo livello (quello che nel caso della tastiera italiana consente di ottenere simboli come la chiocciola e il cancelletto). Ecco cosa si vede a questo proposito nel file '`xkb/rules/xorg.lst`', si tenga presente comunque che non è detto che tutte le alternative funzionino effettivamente:

```
! option
  …
  …
  lv3                   Third level choosers
  lv3:switch            Press Right Control to choose 3rd level
  lv3:menu_switch       Press Menu key to choose 3rd level
  lv3:win_switch        Press any of Win-keys to choose 3rd level
  lv3:lwin_switch       Press Left Win-key to choose 3rd level
  lv3:rwin_switch       Press Right Win-key to choose 3rd level
  …
```

Per fare tutte queste cose durante il funzionamento di X (ammesso che la tastiera sia utilizzabile in qualche modo), si può usare il comando '**setxkbmap**', come nell'esempio seguente, equivalente all'ultima situazione mostrata. Se si usa il comando al di fuori dell'ambiente grafico, occorre aggiungere l'opzione '**-display**', per far sapere al programma dove intervenire.

```
$ setxkbmap -rules xorg -model pc102 -layout "it,fr,de" ↩
↪           -variant basic -option "" ↩
↪           -option "grp:alt_shift_toggle" ↩
↪           -option "lv3:switch" [Invio]
```

Si può comprendere come è stata usata l'opzione '**-option**': la prima volta ha un argomento nullo per azzerare le opzioni; la seconda volta dichiara la combinazione da usare per cambiare gruppo; la terza volta dichiara l'uso del tasto [*Ctrl*] destro per accedere al terzo livello.

In conclusione è bene osservare che **ci sono delle disposizioni di tasti**, come quella canadese ('**ca**') **che definiscono autonomamente più di un gruppo**. In questi casi diventa difficile o impossibile abbinarne altre in modo efficace. Eventualmente, dal momento che si tratta presumibilmente della sovrapposizione della disposizione

statunitense con quella francese, si possono usare queste definizioni affiancate (‘**`Option "XkbLayout" "us,fr"`**’).

Tabella 28.50. Alcuni modelli comuni di tastiera.

Denominazione	Descrizione
`pc101`	La tastiera tradizionale di lingua inglese degli elaboratori «AT», in cui il tasto [*Alt*] destro viene usato normalmente con la stessa funzione di quello sinistro.
`pc102`	La tastiera tradizionale europea degli elaboratori «AT», in cui il tasto [*Alt*] destro serve normalmente per ottenere un terzo livello di simboli.
`pc104`	La tastiera di lingua inglese degli elaboratori «AT» con estensioni per MS-Windows (tre tasti in più), in cui il tasto [*Alt*] destro viene usato normalmente con la stessa funzione di quello sinistro.
`pc105`	La tastiera europea degli elaboratori «AT» con estensioni per MS-Windows (tre tasti in più), in cui il tasto [*Alt*] destro serve normalmente per ottenere un terzo livello di simboli.

Tabella 28.51. Alcune mappe di disposizione dei simboli.

Mappa	Descrizione	Mappa	Descrizione
`al`	Albanese.	`am`	Armena.
`ar`	Araba.	`az`	Azerbagiana.
`be`	Belga.	`bg`	Bulgara.
`br`	Brasiliana.	`bs`	Bosniaca.
`by`	Bielorussa.	`ca`	Canadese.

Mappa	Descrizione	Mappa	Descrizione
cz	Ceca.	de	Tedesca.
dk	Danese.	ee	Estone.
el	Greca	es	Spagnola.
fi	Finlandese.	fr	Francese.
gb	Britannica.	hr	Croata.
hu	Ungherese.	il	Israeliana.
ir	Iraniana.	is	Islandese.
it	Italiana.	mk	Macedone.
mn	Mongola.	mt	Maltese.
nl	Olandese.	no	Norvegese.
pl	Polacca.	pt	Portoghese.
ro	Rumena.	ru	Russa.
se	Svedese.	si	Slovena.
sk	Slovacca.	sr	Serba.
tr	Turca.	ua	Ucraina.
us	Statunitense.	us_intl	Statunitense con accenti morti.

Mappa	Descrizione	Mappa	Descrizione
`en_US`	Inglese generalizzato con estensioni ISO 9995.		

28.6.3 Geometria

«

Tra i file che descrivono i vari tipi di tastiera disponibili con X, ne esiste un gruppo che serve a ottenere una mappa stampata della disposizione dei tasti, con l'aiuto del programma '**`xkbprint`**'.[11] Questo programma, oltre alle opzioni eventuali, prevede l'indicazione obbligatoria delle coordinate dello schermo dal quale si vuole ottenere la mappa della configurazione attuale:

```
xkbprint [opzioni] schermo [file_da_generare]
```

Se non si indica il nome di un file da generare, il programma crea un file nella directory corrente che corrisponde al modello '`server-`***n*** **.** ***estensione***', dove ***n*** descrive le coordinate dello schermo. La tabella successiva descrive alcune opzioni di '**`xkbprint`**'.

Tabella 28.52. Alcune opzioni.

Opzione	Descrizione
`-color` `-mono`	Genera un risultato con o senza colorazione dei tasti.
`-eps`	Genera un risultato in formato EPS, altrimenti si ottiene un formato PostScript.

Opzione	Descrizione
`-fit` `-full`	L'opzione '**`-fit`**', che è predefinita, fa in modo che il risultato sia contenuto in una pagina; con l'opzione '**`-full`**' si ottiene un risultato delle dimensioni normali.
`-label none|name|code|symbols`	Consente di stabilire cosa si vuole vedere sui tasti: nulla, il nome, il codice numerico o il simbolo.
`-ll` ***n***	Richiede di mostrare i simboli a partire dal livello ***n***.
`-o` ***file***	Si tratta di un modo alternativo per indicare il file da generare.
`-pict all|none|common`	Controlla la visualizzazione dei pittogrammi (secondo lo standard ISO 9995-7). Con i tre argomenti alternativi si ottengono rispettivamente: tutti i pittogrammi, nessun pittogramma, solo quelli comuni.

Vengono mostrati alcuni esempi di utilizzo del programma '**`xkbprint`**'; successivamente appaiono alcune figure ottenute dalla configurazione di una mappa italiana a 102 tasti.

- `$` **`xkbprint -color -label symbols -ll 1 -pict all :0 tastiera.ps`** [*Invio*]

  Genera il file PostScript ‘`tastiera.ps`’, con il disegno della tastiera, usando il colori previsti, mostrando i simboli in corrispondenza dei tasti, partendo dal primo livello (predefinito), usando i pittogrammi standard per tutti i tasti con funzioni speciali. La mappa che si ottiene fa riferimento alle coordinate dello schermo ‘**`:0`**’, ovvero ‘**`:0.0`**’.

- `$` **`xkbprint -eps -color -label symbols -ll 1 -pict all :0 tastiera.eps`** [*Invio*]

  Come nell’esempio precedente, ma creando invece il file EPS ‘`tastiera.eps`’.

- `$` **`xkbprint -eps -color -label symbols -ll 1 -pict all :0`** [*Invio*]

  Rispetto all’esempio precedente, non viene indicato il nome del file da creare, pertanto si ottiene il file ‘`server-0.eps`’.

- `$` **`xkbprint -color -label symbols -ll 3 -pict all :0 tastiera.ps`** [*Invio*]

  Questo esempio è simile al primo, dal quale si distingue per specificare la richiesta di utilizzare i simboli a partire dal terzo livello. In pratica, per la tastiera italiana e per altre tastiere europee, si tratta di quei simboli che si ottengono tenendo premuto il tasto [*AltGr*].

- `$` **`xkbprint -color -label code :0 tastiera.ps`** [*Invio*]

  Rispetto al primo esempio, viene generato un disegno contenente i codici numerici dei tasti.

Figura 28.53. ‘**xkbprint -color -label symbols -ll 1 -pict all :0**’

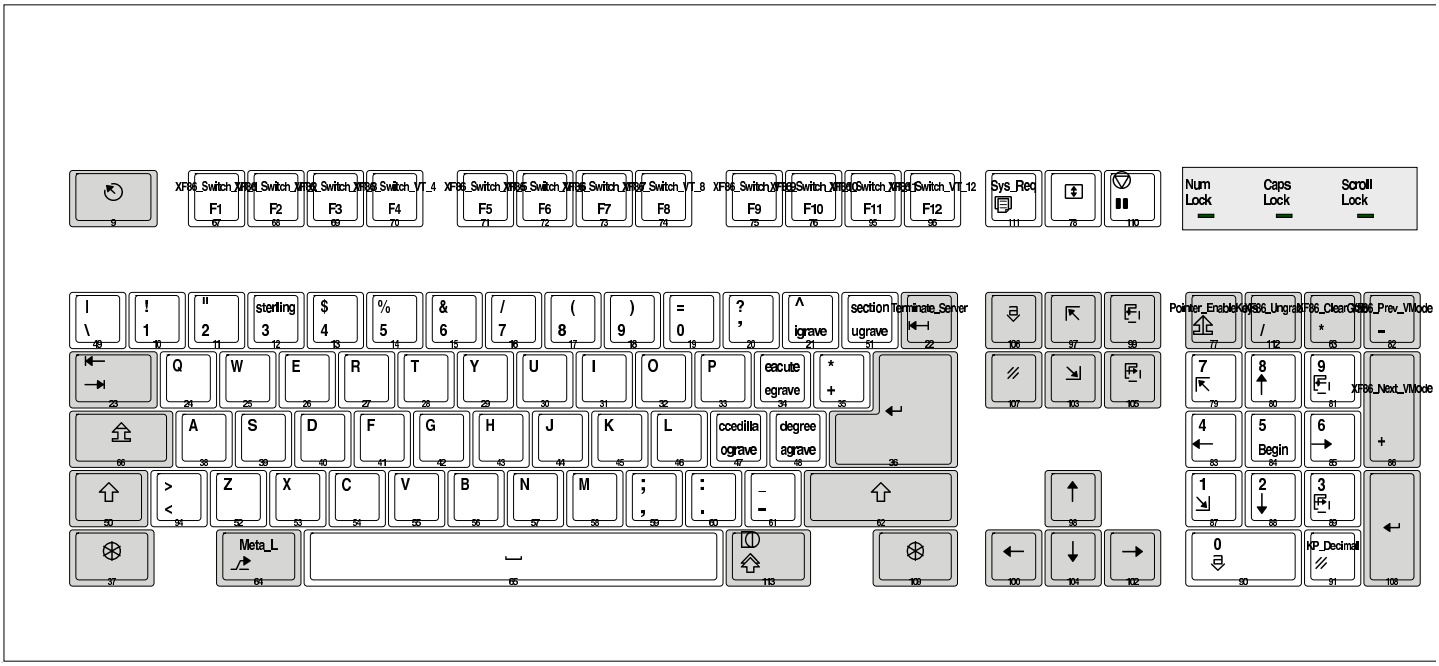


Figura 28.54. ‘**xkbprint -color -label symbols -ll 3 -pict all :0**’

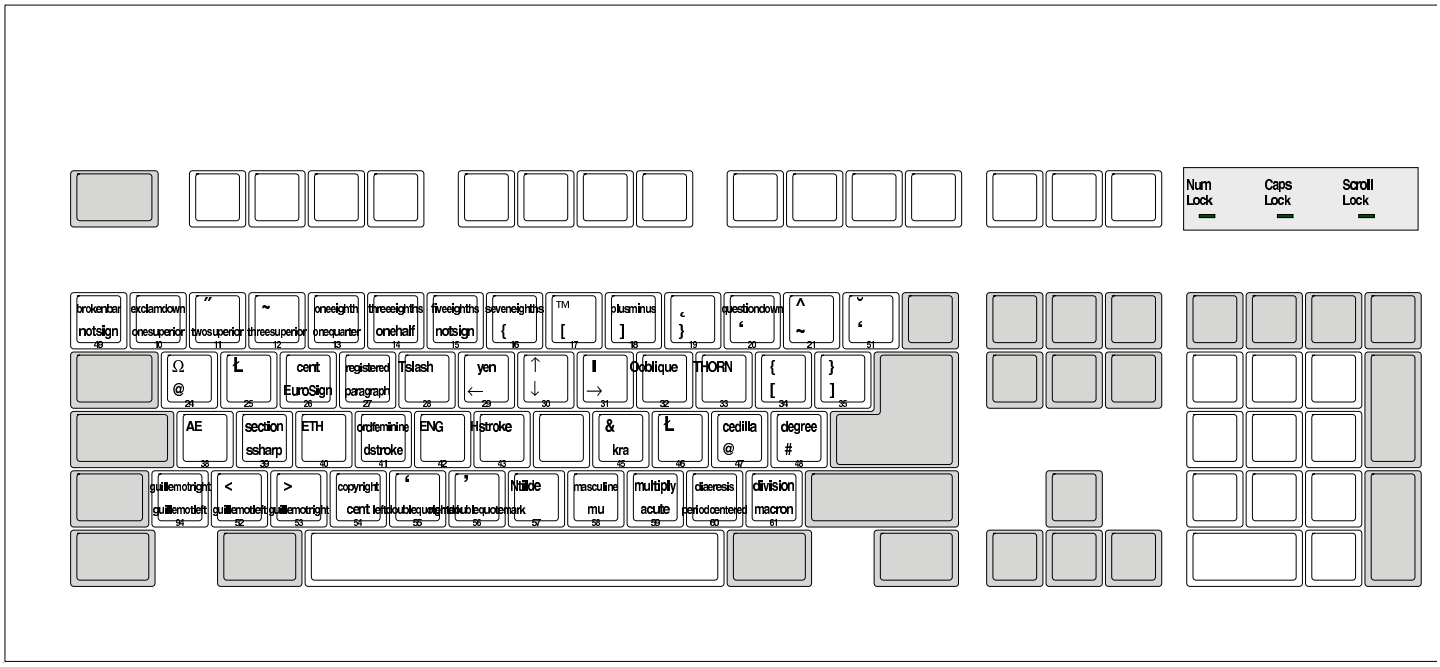
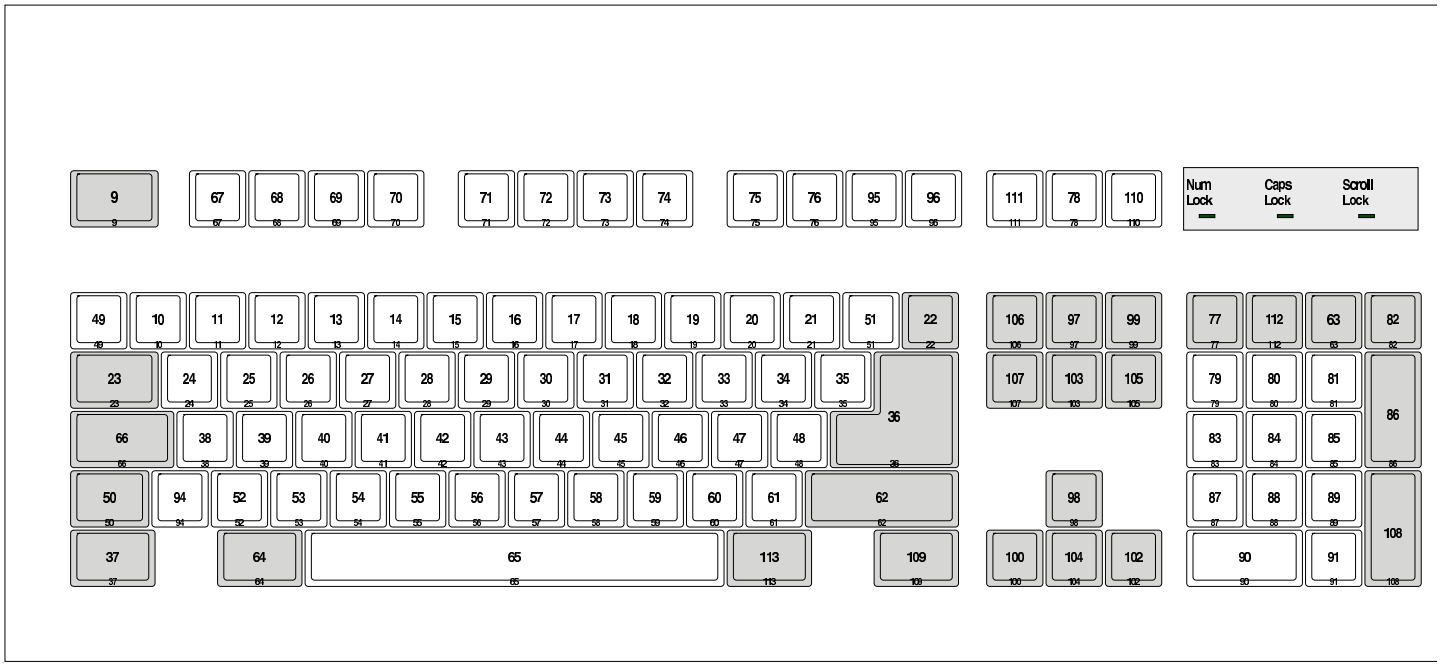


Figura 28.55. ‘**xkbprint -color -label code :0**’



28.7 Configurazione più dettagliata della tastiera e del mouse

XFree86 e X.Org prevedono un tipo di configurazione più dettagliato; eventualmente è possibile intervenire a livello brutale anche attraverso il programma ‘**xmodmap**’. In ogni caso, occorre ricordare che la configurazione della tastiera interferisce anche con quella del mouse; pertanto, un errore fatto da una parte può anche ripercuotersi dall’altra.

Si tenga in considerazione che, a causa della complessità del sistema di gestione della tastiera di X, le indicazioni che appaiono in questo capitolo possono essere imprecise o troppo semplificate.

28.7.1 Componenti della configurazione

Nelle situazioni più comuni, i componenti usati da X.Org per la configurazione della tastiera e del mouse, si trovano a partire dalla directory ‘`/etc/X11/xkb/`’. Per semplicità, qui vengono indicati percorsi del tipo ‘`xkb/…`’. I componenti principali sono cinque:

1. i codici dei tasti (*key code*), annotati all’interno della directory ‘`xkb/keycodes/`’, dove si abbinano i codici numerici della tastiera a dei nomi simbolici, secondo la forma ‘***<xxxx>***’;
2. i tipi, annotati all’interno della directory ‘`xkb/types/`’, dove si definisce il comportamento dei tasti modificatori (quelli che servono a creare delle combinazioni);
3. le mappe di compatibilità, annotate all’interno della directory ‘`xkb/compat/`’, dove si definiscono gli adattamenti da applicare quando i programmi non sono in grado di interpretare correttamente tutte le funzionalità disponibili;
4. i simboli, annotate all’interno della directory ‘`xkb/symbols/`’, dove si definiscono i caratteri associati effettivamente ai nomi simbolici dei tasti;
5. i disegni delle tastiere (*geometry*), annotati all’interno della directory ‘`xkb/geometry/`’, dove si descrivono le tastiere in modo da poterne riprodurre una mappa stampata.

28.7.2 Modificatori

I modificatori sono i tasti o le combinazioni di tasti che servono a cambiare significato ad altri tasti. La configurazione di XFree86 e X.Org distingue tra modificatori reali e modificatori virtuali.

I modificatori reali sono i tasti [*Maiuscole*], [*Fissamaiuscole*], [*Ctrl*] e altri cinque modificatori generici da stabilire, denominati '**Mod1**', '**Mod2**', '**Mod3**', '**Mod4**' e '**Mod5**'. A questi modificatori generici si possono associare tasti come [*Alt*], [*Meta*], [*BlocNum*], che in tal caso sono da considerare dei modificatori virtuali. Eventualmente, un modificatore virtuale può essere costituito in pratica da una combinazione di tasti.

Storicamente, i modificatori presenti praticamente in tutte le tastiere sono solo i primi tre modificatori reali appena descritti; questo spiega la necessità di attribuire gli altri ai modificatori generici.

28.7.3 Configurazione

La configurazione della tastiera può avvenire nel file '`xorg.conf`' in un modo più dettagliato rispetto a quanto si descrive di solito. Si osservi l'esempio seguente:

```
Section "InputDevice"
    Identifier              "Keyboard1"
    Driver                  "Keyboard"
    Option "AutoRepeat"     "500 30"
    Option "XkbKeycodes"    "xorg"
    Option "XkbTypes"       "default"
    Option "XkbSymbols"     "en_US(pc102)+it"
    Option "XkbGeometry"    "pc(pc102)"
    Option "XkbCompat"      "basic+pc+iso9995"
EndSection
```

Quello che si vede rappresenta una configurazione minima per l'utilizzo di una tastiera italiana tipica in un elaboratore x86 con X.Org. Le varie opzioni rappresentano le cinque componenti principali di configurazione, che si trovano all'interno dei file contenuti nelle directory 'xkb/keycodes/', 'xkb/types/', 'xkb/symbols/', 'xkb/geometry/' e 'xkb/compat/' rispettivamente.

In questo caso, l'opzione '**XkbKeycodes**' contiene il valore '**xorg**', che rimanda ai file 'xkb/rules/xorg' e 'xkb/rules/xorg.lst'; nel primo di questi due file c'è un riferimento esplicito alla voce '**xfree86**' per ciò che riguarda i codici della tastiera di un elaboratore x86:

```
! model          =      keycodes
  macintosh_old =      macintosh
  powerpcps2     =      powerpcps2
  pc98           =      xfree98(pc98)
  abnt2          =      xfree86(abnt2)
  jp106          =      xfree86(jp106)
  *              =      xfree86
```

Ciò corrisponde a selezionare il file di configurazione 'xkb/keycodes/xfree86' che contiene l'abbinamento tra codici numerici della tastiera e codici simbolici, come si può vedere dall'estratto seguente:

```
default xkb_keycodes "xfree86" {
    include "xfree86(basic)"
    <BKSL> =  51;
    <LSGT> =  94;
};

xkb_keycodes "basic" {
```

```
    minimum= 8;
    maximum= 255;

    <TLDE> =  49;
    <AE01> =  10;
    <AE02> =  11;
    …
    indicator 1 = "Caps Lock";
    indicator 2 = "Num Lock";
    indicator 3 = "Scroll Lock";

    alias <ALGR> = <RALT>;
};
…
xkb_keycodes "abnt2" {
    include "xfree86(basic)"
    <BKSL> = 94;
    <AC12> = 51;
    <KPPT> = 134;
};
```

Come appena descritto, assegnare il valore '**`xfree86`**' all'opzione '**`XkbKeycodes`**' significa indicare l'uso del file '`xkb/keycodes/xfree86`', il cui contenuto, come si può vedere dall'estratto di esempio, è suddiviso in sezioni. Quando si indica il file, occorrerebbe anche specificare la sezione a cui si fa riferimento; in questo caso la sezione predefinita a essere presa in considerazione è quella denominata nello stesso modo del file, '**`xfree86`**', come si vede all'inizio:

```
default xkb_keycodes "xfree86" {
    include "xfree86(basic)"
…
```

Si può vedere che questa sezione richiama la sezione '**basic**' dello stesso file. In pratica, la configurazione di X ('`/etc/X11/xorg.conf`') avrebbe potuto essere più precisa specificandola così:

```
Section "InputDevice"
    Identifier           "Keyboard1"
    Driver               "Keyboard"
    Option "AutoRepeat"  "500 30"
    Option "XkbKeycodes" "xorg(xfree86)"
    Option "XkbTypes"    "default"
    Option "XkbSymbols"  "en_US(pc102)+it"
    Option "XkbGeometry" "pc(pc102)"
    Option "XkbCompat"   "basic+pc+iso9995"
EndSection
```

L'opzione '**XkbTypes**', che si riferisce alla gestione dei tasti modificatori, contiene il valore '**default**', che corrisponde alla selezione del file '`xkb/types/default`', utilizzando la sezione predefinita con lo stesso nome. Il contenuto del file potrebbe essere questo:

```
default xkb_types "default" {
    include "basic"
    include "pc"
    include "iso9995"
    include "extra"
    include "mousekeys"
};
```

In pratica, viene dichiarato l'uso di altri file della stessa directory secondo la sequenza che si può vedere. La stessa cosa, si può esprimere

nel file di configurazione di X (‘`/etc/X11/xorg.conf`’), come si vede dall’esempio seguente:

```
Section "InputDevice"
    Identifier              "Keyboard1"
    Driver                  "Keyboard"
    Option "AutoRepeat"     "500 30"
    Option "XkbKeycodes"    "xorg(xfree86)"
    Option "XkbTypes"       "basic+pc+iso9995+extra+mousekeys"
    Option "XkbSymbols"     "en_US(pc102)+it"
    Option "XkbGeometry"    "pc(pc102)"
    Option "XkbCompat"      "basic+pc+iso9995"
EndSection
```

Naturalmente si possono rendere esplicite le sezioni; osservando il contenuto dei vari file, si dovrebbe scrivere così:

```
Section "InputDevice"
    Identifier              "Keyboard1"
    Driver                  "Keyboard"
    Option "AutoRepeat"     "500 30"
    Option "XkbKeycodes"    "xorg(xfree86)"
    Option "XkbTypes"       "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"     "en_US(pc102)+it"
    Option "XkbGeometry"    "pc(pc102)"
    Option "XkbCompat"      "basic+pc+iso9995"
EndSection
```

Dopo aver espanso le opzioni precedenti, ‘**`XkbSymbols`**’ comincia a essere più intuitiva: evidentemente viene usata la sezione ‘**`pc102`**’ del file ‘`xkb/symbols/en_US`’ e la sezione predefinita del file ‘`xkb/symbols/it`’. In questo modo si fa riferimento alla mappa inglese con tutte le estensioni del caso, compreso il fatto che si usa una tastiera a 102 tasti, a cui si sovrappone la mappa italiana, che va

a sostituire alcuni tasti. Dall'osservazione dei file, si determina che la sezione predefinita del file '`xkb/symbols/it`' è '**`basic`**':

```
Section "InputDevice"
    Identifier           "Keyboard1"
    Driver               "Keyboard"
    Option "AutoRepeat"  "500 30"
    Option "XkbKeycodes" "xorg(xfree86)"
    Option "XkbTypes"    "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"  "en_US(pc102)+it(basic)"
    Option "XkbGeometry" "pc(pc102)"
    Option "XkbCompat"   "basic+pc+iso9995"
EndSection
```

L'opzione '**`XkbGeometry`**' serve a dichiarare la forma che ha la tastiera. Come ormai si intende, viene indicata la sezione '**`pc102`**' del file '`xkb/geometry/pc`'. Ovviamente è opportuno che la geometria della tastiera sia conforme a quanto dichiarato nell'opzione '**`XkbSymbols`**'.

L'opzione '**`XkbCompat`**' dichiara le «mappe di compatibilità». Come si vede si fa riferimento alle sezioni predefinite dei file '`xkb/compat/basic`', '`xkb/compat/pc`' e '`xkb/compat/iso9995`'. Dall'osservazione dei file si determinano le sezioni predefinite:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes" "xorg(xfree86)"
    Option "XkbTypes"    "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"  "en_US(pc102)+it(basic)"
    Option "XkbGeometry" "pc(pc102)"
    Option "XkbCompat"   "basic(basic)+pc(basic)+↩
↪iso9995(default)"
EndSection
```

Questa modalità di configurazione diventa utile quando si vuole fare qualcosa di diverso da ciò che sarebbe predefinito. Per questo, l'opzione più interessante per gli interventi manuali è '**`XkbSymbols`**'. Per esempio, per attribuire una «variante» alla mappa della tastiera prescelta, occorre trovare presumibilmente una sezione appropriata nel file, che in questo caso è '`xkb/symbols/it`'; così, volendo disabilitare i tasti morti, occorre specificare la sezione '**`nodeadkeys`**':

```
    …
    Option "XkbSymbols"       "en_US(pc102)+it(nodeadkeys)"
    …
```

Con lo stesso criterio, per cambiare la geometria della tastiera, occorre intervenire tra le opzioni di '`xkb/symbols/en_US`' e di '`xkb/geometry/pc`'; in questo caso si passa a una tastiera a 105 tasti:

```
    …
    Option "XkbSymbols"          "en_US(pc105)+it(basic)"
    Option "XkbGeometry"         "pc(pc105)"
    …
```

28.7.3.1 Sintesi della configurazione

Una volta definita in qualche modo la configurazione per la tastiera, si può ottenere una sintesi di tutto ciò che viene coinvolto con l'aiuto del programma '**xkbcomp**',[12] usando l'opzione '**-xkb**':

```
$ xkbcomp -xkb :0 [Invio]
```

In questo modo si richiede di analizzare la configurazione relativa alla tastiera associata allo schermo nelle coordinate '**:0**' (ovvero '**:0.0**'). Ciò che si ottiene in questo caso è il file '`server-0.xkb`', la cui consultazione può essere molto utile per comprendere meglio la configurazione della propria tastiera.

28.7.3.2 Aggiungere nuovi file di configurazione

A parte ogni considerazione sulla difficoltà che comporta la modifica e l'aggiunta di altri file nella struttura che si articola a partire dalla directory '`xkb/`', occorre considerare che i file in questione (e le sezioni in essi contenute) sono riepilogati all'interno di indici; uno per ogni sottodirectory. Per esempio, il file '`xkb/symbols.dir`' si riferisce al contenuto della directory '`xkb/symbols/`'. Ecco un estratto di questo file:

```
-dp----- a------- it(basic)
--p----- a------- it(Sundeadkeys)
--p----- a------- it(sundeadkeys)
--p----- a------- it(nodeadkeys)
```

L'elenco è composto da tre parti: una prima serie di indicatori, una seconda serie e alla fine il file e la sezione a cui si riferiscono. Questi indicatori possono essere teoricamente quelli seguenti:

```
hdp----- amkfg--- nome(sezione)
```

Questi indicatori servono a classificare la sezione che appare a fianco, secondo la logica della tabella seguente:

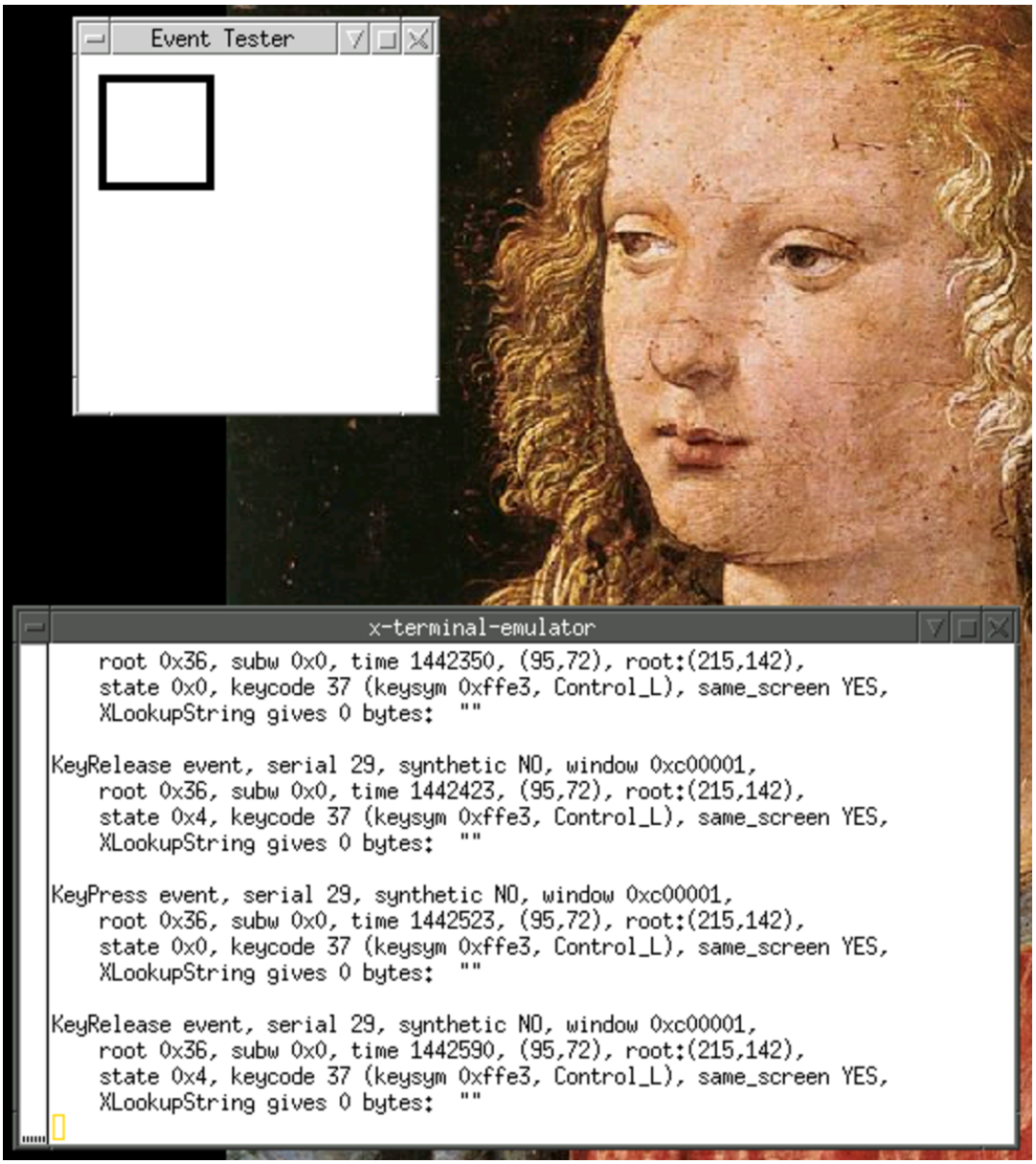
Lettera	Significato mnemonico	Descrizione
`h`	*hidden*	Sezione nascosta da utilizzare soltanto all'interno di altre.
`d`	*default*	Sezione predefinita.
`p`	*partial*	Sezione parziale.
`a`	*alphanumeric*	Funzione riferita alla parte della tastiera che serve a ottenere simboli alfanumerici e non.
`m`	*modifier*	Funzione riferita all'uso dei tasti modificatori.
`k`	*keypad*	Funzione riferita alla gestione della tastiera numerica.
`f`	*function*	Funzione riferita alla gestione di tasti con funzioni speciali.
`g`	*group*	Funzione riferita alla gestione di gruppi alternativi.

28.7.4 Configurazione con «xmodmap»

Nella necessità di modificare o di costruire una mappa personalizzata della tastiera, piuttosto di addentrarsi nei meandri del contenuto delle varie sottodirectory che si articolano a partire da ‘`xkb/`’, può essere più conveniente l’uso del programma ‘**`xmodmap`**’.[13] Per poter usare questo programma è necessario conoscere il codice numerico dei tasti. Per acquisire questa informazione si può utilizzare il programma ‘**`xkbprint`**’, che si avvale delle informazioni contenute nella directory ‘`xkb/geometry/`’, ma in presenza di una tastiera differente dal previsto, occorre usare il programma ‘**`xev`**’.[14]

Il programma ‘**`xev`**’ si avvia da una finestra di terminale e si traduce in un riquadro con sfondo bianco, contenente un piccolo rettangolo. Quando la sua finestra è in primo piano o comunque quando è attiva, le azioni che si compiono con la tastiera o con il mouse, vengono descritte nella finestra di terminale da cui il programma è stato avviato. La figura successiva dà un’idea di questo comportamento.

Figura 28.70. Il programma '**xev**' in funzione.



Per esempio, premendo il tasto [*Ctrl*] sinistro di una tastiera comune si dovrebbe ottenere un risultato simile a quello seguente:

```
KeyPress event, serial 29, synthetic NO, window 0x1000001,
    root 0x36, subw 0x0, time 2064092, (336,380), root:(339,399),
    state 0x0, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyRelease event, serial 29, synthetic NO, window 0x1000001,
    root 0x36, subw 0x0, time 2064203, (336,380), root:(339,399),
    state 0x4, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""
```

Da questo esempio si comprende che il codice del tasto [*Ctrl*] sinistro è il numero 37.

Con queste informazioni si può usare il programma '**xmodmap**' per sostituire la funzione di qualche tasto, oppure per ridefinire completamente la tastiera.

28.7.4.1 Utilizzo di «xmodmap»

Il programma '**xmodmap**' si avvale normalmente di un file di configurazione, che, se non lo si specifica espressamente, corrisponde a '`.xmodmap`' contenuto nella directory personale dell'utente. Per questa ragione il programma può fare a meno di argomenti nella riga di comando:

```
xmodmap [opzioni] [file_di_configurazione]
```

Tabella 28.72. Alcune opzioni.

Opzione	Descrizione
`-display` ***schermo***	Permette di specificare a quale servente grafico fare riferimento.
`-grammar`	Genera un promemoria della sintassi da usare per la configurazione.

Opzione	Descrizione
`-n`	Simula l'operazione, senza eseguirla veramente.
`-e` ***espressione***	Esegue l'espressione fornita, che in pratica è l'equivalente di una direttiva di configurazione. Si può usare questa opzione più volte.
`-pm`	*print modifier* Emette la configurazione attuale dei modificatori.
`-pk`	*print keymap* Emette la configurazione attuale della mappa della tastiera.
`-pke`	*print keymap expression* Emette la configurazione attuale della mappa della tastiera, usando un formato compatibile con quello necessario per la configurazione.
`-pp`	*print pointermap* Emette la configurazione attuale del puntatore grafico.

Segue la descrizione di alcuni esempi di interrogazione sullo stato attuale della configurazione, quando il servente grafico è stato avviato avendo il file di configurazione di X ('`/etc/X11/xorg.conf`') con la sezione '**`InputDevice`**' organizzata così:

```
Section "InputDevice"
    Identifier                  "Keyboard1"
    Driver                      "Keyboard"
    Option "AutoRepeat"         "500 30"
    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"         "en_US(pc102)+it(basic)"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic(basic)+pc(basic)+↩
↪iso9995(default)"
EndSection
```

- `$` **`xmodmap -grammar`** [*Invio*]

```
xmodmap accepts the following input expressions:

    pointer = default             reset pointer buttons to default
    pointer = NUMBER ...          set pointer button codes
    keycode NUMBER = [KEYSYM ...] map keycode to given keysyms
    keysym KEYSYM = [KEYSYM ...]  look up keysym and do a keycode operation
    clear MODIFIER                remove all keys for this modifier
    add MODIFIER = KEYSYM ...     add the keysyms to the modifier
    remove MODIFIER = KEYSYM ...  remove the keysyms from the modifier

where NUMBER is a decimal, octal, or hex constant; KEYSYM is a valid
Key Symbol name; and MODIFIER is one of the eight modifier names:  Shift,
Lock, Control, Mod1, Mod2, Mod3, Mod4, or Mod5.  Lines beginning with
an exclamation mark (!) are taken as comments.  Case is significant except
for MODIFIER names.

Keysyms on the left hand side of the = sign are looked up before any changes
are made; keysyms on the right are looked up after all of those on the left
have been resolved.  This makes it possible to swap modifier keys.
```

- `$` **`xmodmap -pm`** [*Invio*]

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

```
shift       Shift_L (0x32),  Shift_R (0x3e)
lock        Caps_Lock (0x42)
control     Control_L (0x25),  Control_R (0x6d)
mod1        Alt_L (0x40)
mod2        Num_Lock (0x4d)
mod3        Mode_switch (0x71)
mod4        Super_L (0x73)
mod5        Scroll_Lock (0x4e)
```

- `$` **xmodmap -pk** [*Invio*]

```
There are 4 KeySyms per KeyCode; KeyCodes range from 8 to 255.
…
  8
  9  0xff1b (Escape)
 10  0x0031 (1)      0x0021 (exclam)     0x00b9 (onesuperior)    0x00a1 (exclamdown)
 11  0x0032 (2)      0x0022 (quotedbl)   0x00b2 (twosuperior)    0xfe59 (dead_doubleacute)
 12  0x0033 (3)      0x00a3 (sterling)   0x00b3 (threesuperior)  0xfe53 (dead_tilde)
…
115  0xffeb (Super_L)
116  0xff20 (Multi_key)
117  0xff67 (Menu)
118
119
…
254
255
```

- `$` **xmodmap -pke** [*Invio*]

```
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
keycode  12 = 3 sterling threesuperior dead_tilde
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
```

```
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

- `$` **`xmodmap -pp`** [*Invio*]

```
There are 3 pointer buttons defined.

    Physical        Button
     Button          Code
        1              1
        2              2
        3              3
```

28.7.4.2 Configurazione della tastiera numerica

In presenza di una tastiera comune, con la quale si intende sfruttare il blocco dei tasti numerici esclusivamente per questo, eliminando del tutto la possibilità di usare quei tasti anche per il movimento del cursore e altro, si possono ridefinire le funzioni associate ai tasti, senza distinzione tra i livelli. Per cominciare occorre ottenere la configurazione attuale:

`$` **`xmodmap -pke`** [*Invio*]

```
…
keycode  79 = KP_Home KP_7
keycode  80 = KP_Up KP_8
keycode  81 = KP_Prior KP_9
…
keycode  83 = KP_Left KP_4
keycode  84 = KP_Begin KP_5
```

```
keycode  85 = KP_Right KP_6
…
keycode  87 = KP_End KP_1
keycode  88 = KP_Down KP_2
keycode  89 = KP_Next KP_3
keycode  90 = KP_Insert KP_0
keycode  91 = KP_Delete KP_Decimal
…
```

Con questi dati si può costruire un file di configurazione contenente le righe seguenti:

```
keycode  79 = KP_7
keycode  80 = KP_8
keycode  81 = KP_9
keycode  83 = KP_4
keycode  84 = KP_5
keycode  85 = KP_6
keycode  87 = KP_1
keycode  88 = KP_2
keycode  89 = KP_3
keycode  90 = KP_0
keycode  91 = KP_Decimal
```

Queste righe possono essere inserite all'interno del file '`~/.xmodmap`', oppure in un altro file da indicare espressamente nella riga di comando di '**`xmodmap`**'. In alternativa si può usare anche l'opzione '**`-e`**', in questo modo:

```
$ xmodmap -e "keycode 79 = KP_7" [Invio]

$ xmodmap -e "keycode 80 = KP_8" [Invio]

$ xmodmap -e "keycode 81 = KP_9" [Invio]
```

```
$ xmodmap -e "keycode 83 = KP_4" [Invio]

$ xmodmap -e "keycode 84 = KP_5" [Invio]

$ xmodmap -e "keycode 85 = KP_6" [Invio]

$ xmodmap -e "keycode 87 = KP_1" [Invio]

$ xmodmap -e "keycode 88 = KP_2" [Invio]

$ xmodmap -e "keycode 89 = KP_3" [Invio]

$ xmodmap -e "keycode 90 = KP_0" [Invio]

$ xmodmap -e "keycode 91 = KP_Decimal" [Invio]
```

28.7.4.3 Configurazione dei tasti del mouse

Normalmente i tasti del mouse sono configurati in modo da avere il primo tasto, ovvero quello premuto più spesso, sotto al dito indice della mano destra. Una persona che volesse usare il mouse con la mano sinistra dovrebbe ridefinire la sequenza dei tasti con una configurazione per '**xmodmap**' pari all'esempio seguente:

```
pointer = 3 2 1
```

Usando l'opzione '**-e**' basta il comando seguente:

```
$ xmodmap -e "pointer = 3 2 1" [Invio]
```

Si può verificare l'inversione con l'opzione '**-pp**':

```
$ xmodmap -pp [Invio]
```

```
There are 3 pointer buttons defined.

    Physical        Button
     Button          Code
        1              3
        2              2
        3              1
```

28.7.4.4 Riconfigurazione della mappa della tastiera

«

La riconfigurazione della mappa della tastiera può essere eseguita facilmente se si dispone già di una mappa abbastanza simile a quella che si vuole ottenere. Per prima cosa occorre osservare l'associazione dei tasti modificatori:

```
$ xmodmap -pm [Invio]

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

shift       Shift_L (0x32),  Shift_R (0x3e)
lock        Caps_Lock (0x42)
control     Control_L (0x25),  Control_R (0x6d)
mod1        Alt_L (0x40)
mod2        Num_Lock (0x4d)
mod3        Mode_switch (0x71)
mod4        Super_L (0x73)
mod5        Scroll_Lock (0x4e)
```

Per riprodurre la stessa cosa attraverso direttive di configurazione di '**xmodmap**', occorrono le righe seguenti:

```
clear shift
clear lock
```

```
clear control
clear mod1
clear mod2
clear mod3
clear mod4
clear mod5
add shift = Shift_L Shift_R
add lock = Caps_Lock
add control = Control_L Control_R
add mod1 = Alt_L
add mod2 = Num_Lock
add mod3 = Mode_switch
add mod5 = Scroll_Lock
```

Successivamente, si può riprodurre la mappa attuale, direttamente in forma di direttive di configurazione per '**xmodmap**', con l'opzione '**-pke**':

```
$ xmodmap -pke [Invio]

keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

Per ottenere quasi la stessa cosa, basta assemblare le due parti in un file da dare in pasto a '**xmodmap**':

```
clear shift
clear lock
clear control
clear mod1
clear mod2
clear mod3
clear mod4
clear mod5
add shift = Shift_L Shift_R
add lock = Caps_Lock
add control = Control_L Control_R
add mod1 = Alt_L
add mod2 = Num_Lock
add mod3 = Mode_switch
add mod5 = Scroll_Lock
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

Naturalmente, una volta verificato che tutto funziona (quasi) come prima, si può fare qualche modifica. Dovrebbe essere evidente che

ogni direttiva '**keycode**' ha alla destra del segno '**=**' la sequenza dei quattro livelli per il tasto corrispondente.

È importante osservare che l'associazione dei modificatori non è uniforme tra le definizioni delle tastiere dei vari paesi. Per esempio, non è detto che la funzione per il passaggio al terzo livello sia associata al modificatore virtuale '**Mode_switch**', come invece avviene in questo caso; inoltre non è detto che sia associata al modificatore reale '**Mod3**'. Purtroppo, l'associazione dei modificatori non può essere cambiata, se non seguendo attentamente le caratteristiche della configurazione contenuta a partire da '`xkb/`'.

Se si dispone di una tastiera che, pur risultando compatibile dal punto di vista elettronico, mostra di avere un'associazione differente dei codici numerici, occorre usare il programma '**xev**' per scoprire l'abbinamento corretto, quindi si può cercare di adattare una mappa già esistente, secondo la modalità già mostrata. Naturalmente ciò richiede un lavoro più lungo, tenendo conto che la prima cosa da associare correttamente sono proprio i modificatori che si vogliono usare. Tornando all'esempio già mostrato, i vari tasti [*Maiuscole*], [*Fissamaiuscole*], [*Ctrl*], [*Alt*],... sono associati così:

```
keycode  50 = Shift_L
keycode  62 = Shift_R
keycode  66 = Caps_Lock
keycode  37 = Control_L
keycode 109 = Control_R
keycode  64 = Alt_L Meta_L
keycode  77 = Num_Lock Pointer_EnableKeys
keycode 113 = Mode_switch
keycode  78 = Scroll_Lock
```

28.8 Metodi di inserimento intelligente con SCIM

«

La tastiera per elaboratore nasce come evoluzione di quella della macchina da scrivere, secondo una logica «occidentale», dove i caratteri tipografici sono in un numero molto limitato. Per scrivere i caratteri di lingue che ne annoverano una quantità maggiore, occorre un sistema alternativo di inserimento, che di solito si basa su una forma di traslitterazione a partire dall'alfabeto latino; per esempio, nel caso della lingua cinese, di solito si attribuiscono dei nomi ai caratteri.

Con X, per realizzare l'obiettivo della scrittura con caratteri di lingue orientali che utilizzano molti simboli, si usa normalmente un programma che viene attivato dalle applicazioni che richiedono l'inserimento, attraverso una combinazione di tasti che viene interpretata dalle applicazioni stesse. In pratica, se la configurazione locale è attinente, le applicazioni tendono a dare un significato particolare a certe combinazioni di tasti, con le quali attivano o disattivano il programma di inserimento guidato dei caratteri di una certa lingua.

28.8.1 SCIM

SCIM,[15] ovvero *Smart common input method*, viene definita una «piattaforma» per diversi metodi di inserimento intelligente. SCIM è principalmente un servente che viene interpellato dai programmi grafici quando richiesto attraverso la combinazione di tasti [*Ctrl Spazio*]. Questo servente, che costituisce la piattaforma SCIM, si avvale di moduli specifici per ogni metodo di inserimento. Questi moduli si chiamano genericamente IMEngine (*Input method engine*) e a seconda della presenza di questo o di quel modulo, SCIM è in grado di offrire il metodo di inserimento intelligente relativo.

Di solito, la piattaforma SCIM viene avviata come demone a uso di un utente singolo (se più utenti accedono allo stesso elaboratore, ognuno avvia la sua copia del demone):

```
scim [opzioni]
```

In condizioni normali, si usa solo l'opzione '**-d**', con la quale si ottiene proprio il funzionamento del programma '**scim**' in qualità di demone, sullo sfondo:

```
$ scim -d[Invio]

  Launching a SCIM daemon with Socket FrontEnd...
  Loading simple Config module ...
  Creating backend ...
  Loading socket FrontEnd module ...
  Starting SCIM as daemon ...
  Launching a SCIM process with x11...
  Loading socket Config module ...
  Creating backend ...
  Loading x11 FrontEnd module ...
```

```
GTK Panel of SCIM 1.4.4
SCIM has been successfully launched.
Smart Common Input Method 1.4.4
```

Come si può intuire dall'esempio, il demone '**scim**' va avviato nell'ambito del sistema grafico X, eventualmente da una finestra di terminale, se non è possibile fare di meglio. Tuttavia, occorre predisporre prima delle variabili di ambiente:

```
XMODIFIERS='@im=SCIM'
GTK_IM_MODULE="scim"
QT_IM_MODULE="scim"
export XMODIFIERS
export GTK_IM_MODULE
export QT_IM_MODULE
#
scim -d
```

28.8.2 Attivazione del metodo di inserimento intelligente

Una volta definite le variabili di ambiente e il demone '**scim**' come descritto nella sezione precedente, alcuni programmi associano senza altre preoccupazioni la combinazione di tasti [*Ctrl Spazio*] all'emersione di un programma frontale con il quale si può dichiarare il tipo di metodo di inserimento preferito (in base ai moduli disponibili).

Figura 28.90. Il pannellino di selezione per il metodo di inserimento intelligente.

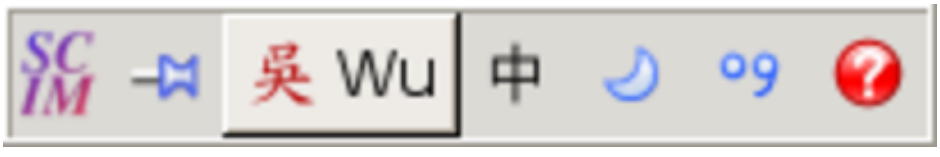
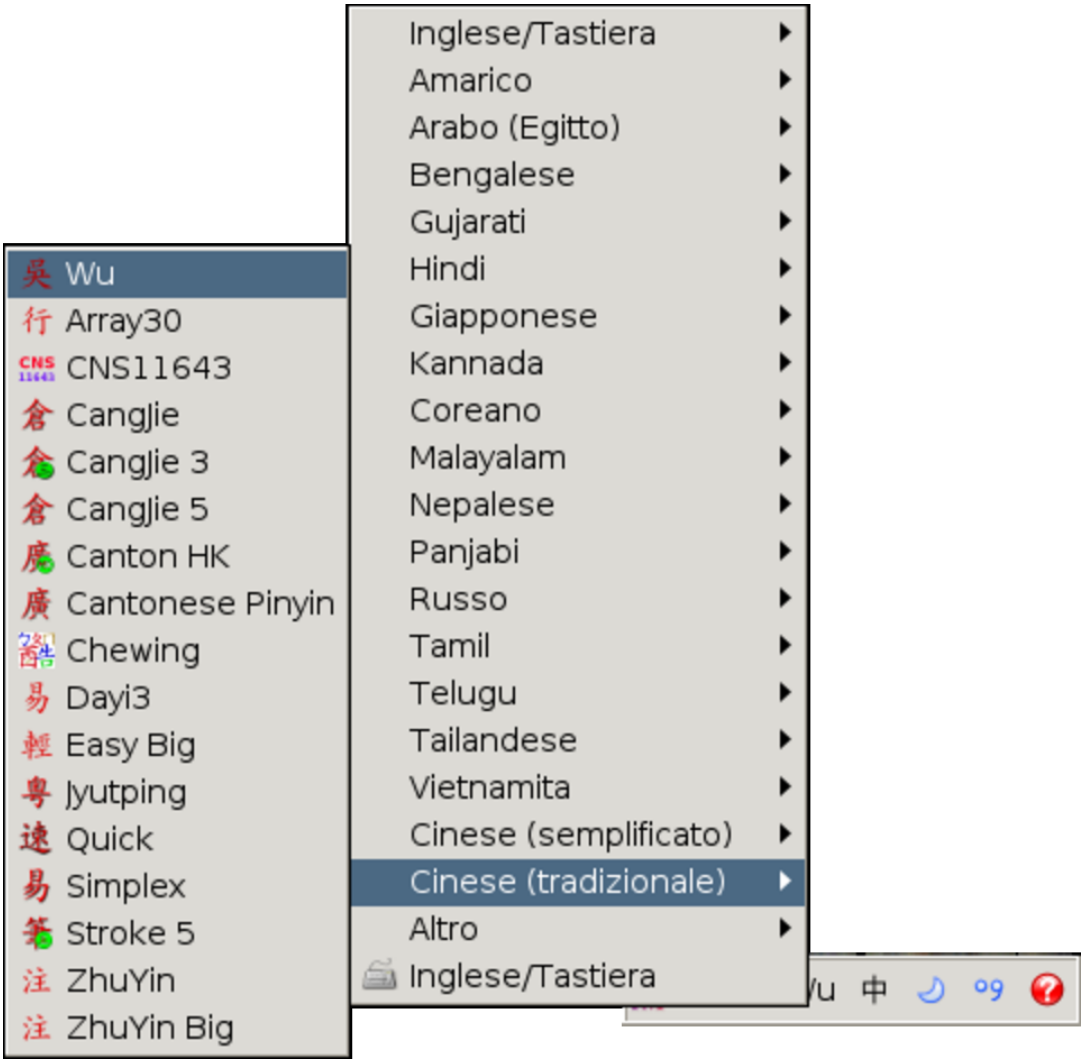


Figura 28.91. Il menù di selezione del metodo di inserimento intelligente.



28.8.3 La configurazione locale

Alcuni programmi, alla pressione della combinazione [*Ctrl Spazio*] attivano sempre il pannellino di controllo di SCIM, ammesso che il demone '**scim**' sia disponibile; altri programmi lo fanno solo se la configurazione locale è appropriata, secondo il loro punto di vista.

Per esempio, LibreOffice è uno di quei programmi che necessitano di una configurazione locale «orientale». Esempi di tali configurazioni sono '**zh_CN.UTF-8**' e '**ja_JP.UTF-8**'; naturalmente si richiede

l'uso dell'insieme di caratteri universale, attraverso la codifica UTF-8.

Come è noto, per impostare il linguaggio è sufficiente assegnare un valore alla variabile di ambiente ***LANG***, per farlo ereditare in modo predefinito a tutte le altre variabili ***LC_*****∗**. Tuttavia, se lo si preferisce, al fine di attivare le funzioni di SCIM, è sufficiente che la variabile di ambiente ***LC_CTYPE*** sia impostata in questo modo. Si vede l'esempio della configurazione cinese comune, in entrambi i casi (nel secondo caso la configurazione locale predefinita è quella della lingua italiana, con l'eccezione della variabile ***LC_CTYPE***):

```
LANG=zh_CN.UTF-8
```

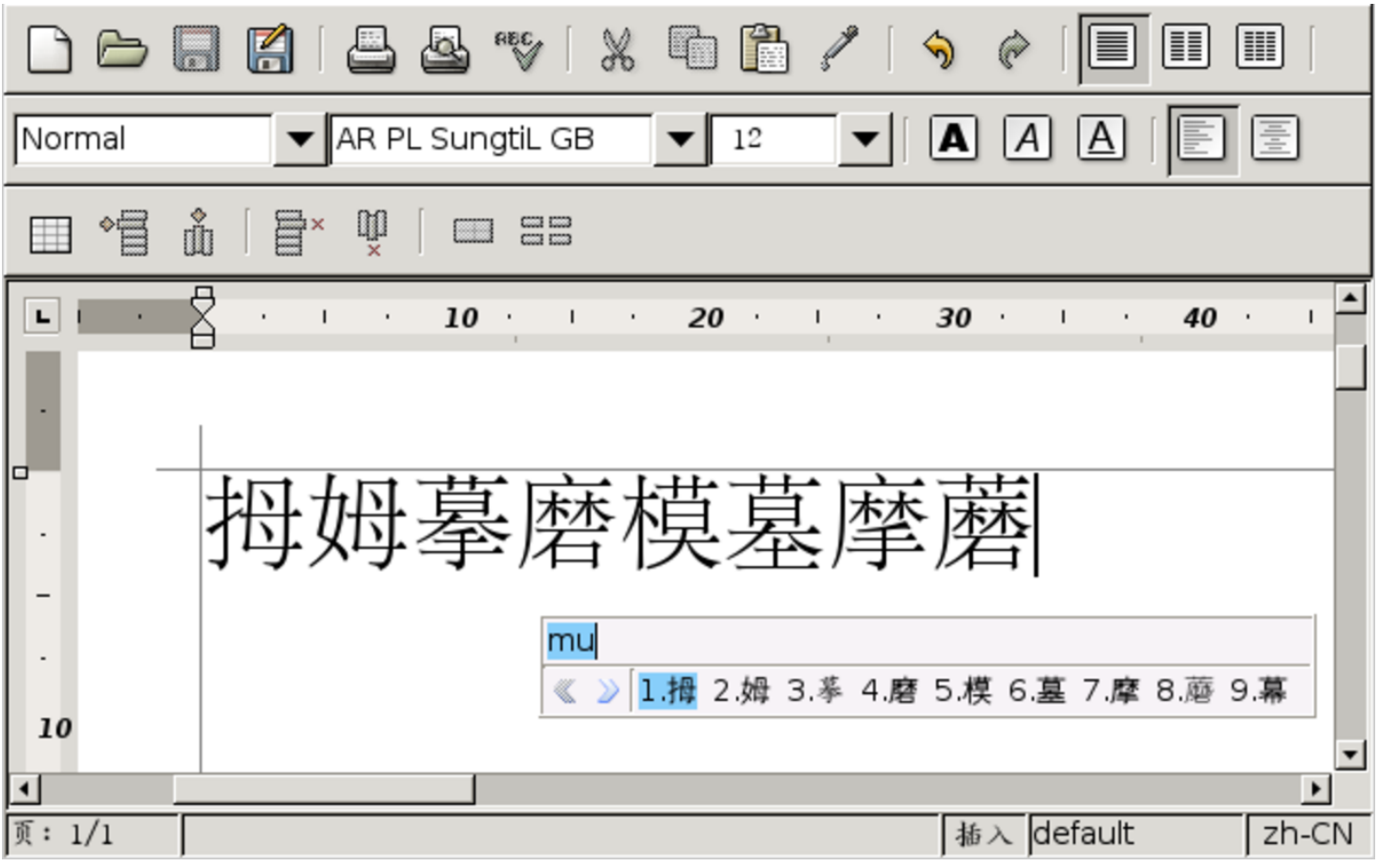
```
LANG=it_IT.UTF-8
LC_CTYPE=zh_CN.UTF-8
```

L'utilizzo della variabile di ambiente ***LC_CTYPE***, con una configurazione locale differente da quella complessiva, ha però degli effetti collaterali. In particolare, supponendo di utilizzare proprio la configurazione dell'ultimo esempio mostrato (in generale lingua italiana, mentre ***LC_CTYPE*** è impostata per il cinese), si ottiene sì un ambiente che comunica in italiano, consentendo l'uso di una modalità di inserimento cinese (e probabilmente anche di altre lingue), ma l'inserimento di valori numerici potrebbe richiedere l'uso del punto per separare la parte intera da quella decimale (mentre in italiano si richiede l'uso della virgola).

28.8.4 Utilizzo

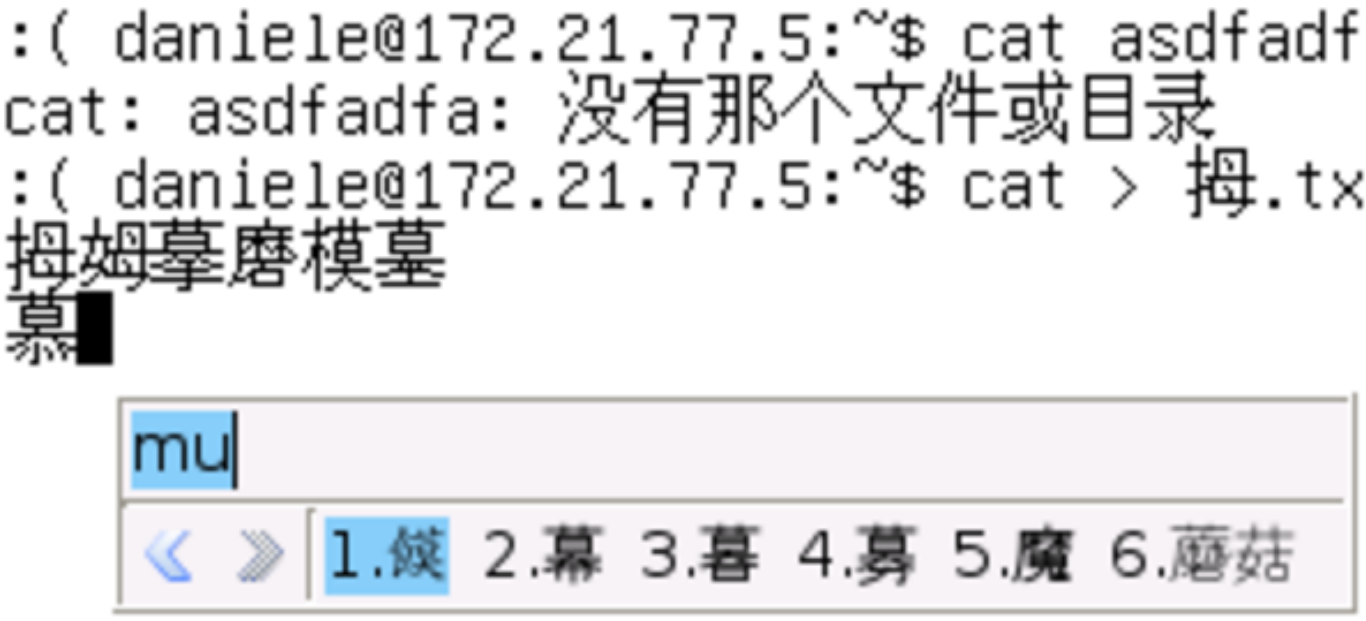
Per inserire effettivamente i caratteri desiderati, dopo avere selezionato il metodo di inserimento, occorre passare all'applicazione nella quale va inserito il testo e si procede digitando la traslitterazione dei caratteri voluti. A seconda del metodo di inserimento, si ottiene un meccanismo di completamento, eventualmente con la proposta di un certo numero di caratteri alternativi da scegliere, come si vede nella figura successiva. Per selezionare un carattere in questo modo, è sufficiente premere il numero corrispondente, oppure si può agire con il mouse, in modo abbastanza intuitivo.

Figura 28.94. Esempio di inserimento di un testo in cinese: inserendo la sequenza «mu», si ottiene un elenco di caratteri alternativi che possono corrispondere.



Si osservi che, quando si usa un programma di scrittura, è necessario che il tipo di carattere scelto contenga i simboli che si vogliono inserire. Solo i programmi più evoluti provvedono da soli a cambiare il tipo di carattere quando quello originario non li contiene.

Figura 28.95. Esempio di inserimento di un testo in cinese con una finestra di terminale.



28.9 Gestore di finestre: Fvwm

Il gestore di finestre, o *window manager* (WM), è quel programma cliente, che si occupa di incorniciare le superfici degli altri programmi clienti, di gestire la messa a fuoco, il passaggio da un programma all'altro e di altre funzioni di contorno. Anche se apparentemente non sembra molto, il gestore di finestre è in grado di cambiare la faccia e il funzionamento operativo del sistema X.

Alcuni gestori di finestre consentono di utilizzare una superficie maggiore di quella che si vede sullo schermo. Si parla in questi casi di gestori di finestre con superficie grafica virtuale, ovvero di *vir-*

tual window manager (VWM). Di solito, per passare da una zona all'altra della superficie grafica virtuale si utilizza la combinazione [*Ctrl freccia...*] nella direzione in cui ci si vuole spostare, oppure si utilizza il mouse all'interno di una tabellina riassuntiva di tutta la superficie grafica virtuale.

Volendo, a puro titolo didattico, si può utilizzare X senza un gestore di finestre:

```
$ xinit /usr/bin/X11/xterm -geometry =50x10+10+10 [Invio]
```

Oppure:

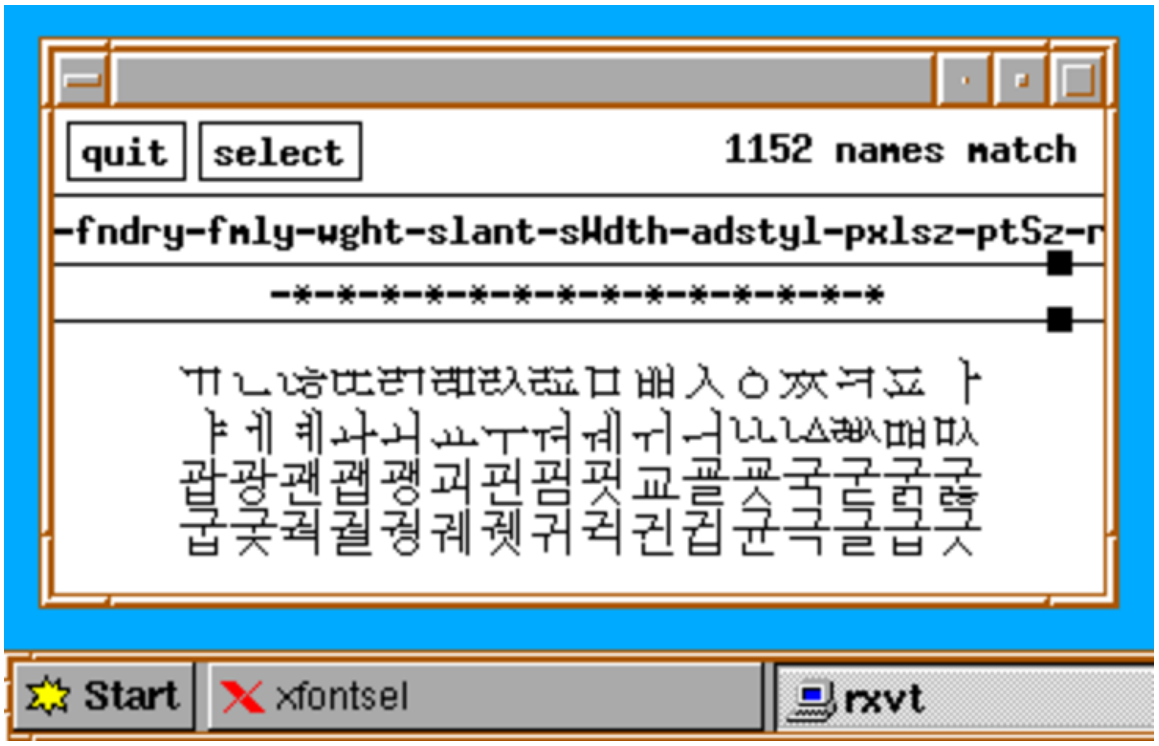
```
$ startx /usr/bin/X11/xterm -geometry =50x10+10+10 [Invio]
```

La figura 28.96 mostra il risultato di questo comando. Quando termina l'esecuzione del programma '**xterm**', '**xinit**' fa terminare il funzionamento del servente.

Figura 28.96. Il servente X avviato senza un gestore di finestre.

A titolo di esempio viene mostrato il funzionamento e la configurazione di Fvwm[16], un gestore di finestre molto semplice e facile da adattare.

Figura 28.97. Il programma '**xfontsel**' eseguito all'interno di Fvwm.



Per fare in modo che, attraverso lo script '**startx**', si avvii automaticamente il gestore di finestre Fvwm, occorre ricordare di modificare il proprio script '`~/.xinitrc`', inserendovi la chiamata all'eseguibile '**fvwm**' (o '**fvwm2**', a seconda di come viene denominato l'eseguibile dalla propria distribuzione). Generalmente è sufficiente avviare il gestore di finestre, senza altri programmi accessori:

```
# start some nice programs

exec fvwm
```

La configurazione generale di Fvwm risiede normalmente nel file '`/etc/X11/fvwm/system.fvwmrc`'; eventualmente, ogni utente può definire la propria configurazione personale, nel file '`~/.fvwmrc`', che in tal caso si sostituisce a quella generale.

Anche se esiste sostanzialmente una sola versione di Fvwm che continui a essere sviluppata, sono esistite diverse varianti del programma con configurazioni non compatibili tra di loro. Per questo, a seconda della distribuzione GNU utilizzata, può darsi che i file di configurazione debbano includere il numero più significativo della versione. Nel caso particolare di Fvwm 2.∗, i file potrebbero essere ‘`system.fvwm2rc`’ e ‘`.fvwm2rc`’.

Il file di configurazione predefinito potrebbe essere molto complesso, ma adeguatamente commentato in modo da guidare chi desidera modificarlo. In generale, non è conveniente personalizzare tutto. Di sicuro è necessario sistemare i menù, mentre il resto può rimanere come si trova. Il file *allegati/system.fvwmrc* rappresenta un esempio di configurazione completa di Fvwm 2.*, ridotta all'essenziale, con la presenza di una barra di avvio simile a quella di MS-Windows 95/98.

28.10 Accorgimenti per la costruzione di un menù

Quando si vuole realizzare un menù per un gestore di finestre, o per un programma specifico da usare nell'ambiente grafico, possono essere utili degli script che si vogliono annotare in questo capitolo.

Naturalmente, il linguaggio usato per la costruzione del menù potrebbe includere già delle direttive di controllo, tali da rendere superflui alcuni o tutti gli script che vengono proposti qui; pertanto ogni cosa va usata se effettivamente ne esiste la convenienza.

Gli script proposti, che si basano sulla disponibilità di una shell POSIX (o quasi), sono semplificati al massimo; sta poi alla sensibilità di ognuno estenderli secondo le proprie preferenze. Si osservi an-

che che, se incorporati nelle direttive della configurazione del menù, vanno forse modificati in qualche modo per proteggere alcuni simboli; inoltre, potrebbe essere necessario tradurli in istruzioni disposte su una sola riga.

28.10.1 Verifica della disponibilità

«

Quando si realizza un menù per facilitare l'avvio di alcuni programmi, può essere utile verificare prima la loro esistenza effettiva, in modo da avvisare l'utente in caso contrario. Infatti, durante il funzionamento in modalità grafica si perdono normalmente le segnalazioni di errore fornite dalla shell o dal sistema operativo.

L'esempio seguente riguarda l'avvio del programma '**gnumeric**', collocato precisamente nella directory '`/usr/bin/`'; se il file dovesse mancare o se semplicemente non dovesse risultare eseguibile, si otterrebbe una finestra di terminale con un avvertimento, che rimane evidente per pochi secondi:

```
if [ -x /usr/bin/gnumeric ]
then
    gnumeric
else
    xterm -e "echo Sorry, gnumeric is not available. ; sleep 15"
fi
```

Segue lo stesso esempio, facendo uso però di '**xmessage**', al posto di '**xterm**':

```
if [ -x /usr/bin/gnumeric ]
then
    gnumeric
else
    xmessage -timeout 15 "Sorry, gnumeric is not available."
fi
```

28.10.2 Verificare che un programma non sia già in funzione

Quando la configurazione dell'ambiente grafico non prevede la presenza di segnalazioni che avvisano dell'avvio in corso di un programma, l'utente inesperto può essere indotto a ritentare l'avvio di qualcosa che sembra non dare segni di vita. Questo tipo di problema si aggrava quando l'accesso alla memoria di massa è relativamente lento e finisce con il rallentare ancora di più l'avvio del programma stesso, salvo alla fine ritrovarsi in funzione tutte le copie richieste in sequenza.

Quando un programma consente al proprio interno di avviare altre copie di se stesso, o di aprire più documenti distinti in schede separate, può essere utile uno script che verifica lo stato dei processi elaborativi appartenenti allo stesso utente:

```
if ps x | grep -v grep | grep "[0-9] gnumeric" 2>&1 > /dev/null
then
    xterm -e "echo Already running ; sleep 10"
else
    gnumeric
fi
```

La stessa cosa, facendo uso però di '**xmessage**', al posto di '**xterm**':

```
if ps x | grep -v grep | grep "[0-9] gnumeric" 2>&1 > /dev/null
then
    xmessage -timeout 10 "Already running"
else
    gnumeric
fi
```

28.10.3 Informare della fase di avvio del programma

La fase di avvio di un programma può essere resa evidente attraverso l'uso del comando '**ps rx**', con il quale si evidenziano i processi residenti in memoria (quindi i processi attivi), appartenenti all'utente. Quando il processo elaborativo raggiunge la quiete e non si trova più a essere residente, di solito ha già mostrato la finestra grafica che lo riguarda. L'esempio seguente si riferisce all'avvio di AbiWord:

```
xterm -e "echo Loading abiword... ; ↩
↪sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] abiword\" ; ↩
↪do sleep 2 ; ↩
↪done" & ↩
↪abiword
```

Come si può osservare, la maggior parte dello script è inserita nel comando avviato da '**xterm**', perché al termine del ciclo di controllo il terminale deve chiudersi.

Si osservi che la verifica fatta con '**ps**' potrebbe dover includere altri programmi, se quello che si avvia, prima di potersi presentare ne deve avviare. Questo succede spesso con i programmi per Gnome o per KDE. Segue un altro esempio riferito all'avvio di Gnumeric:

```
xterm -e "echo Loading gnumeric... ; ↩
↪sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] gnumeric\" ↩
↪|| ps rx | grep -v grep | grep \"gconfd\" ↩
↪|| ps rx | grep -v grep | grep \"bonobo-activation-server\" ; ↩
↪do sleep 2 ; ↩
↪done" & ↩
↪gnumeric
```

Eventualmente, si può decidere di considerare qualunque programma che non sia in quiete, anche se può succedere che la finestra del terminale non si chiuda più, per la presenza di un processo

impegnativo indipendente:

```
xterm -e "echo Loading gnumeric... ; sleep 2 ;
while ps rx | grep -v grep ;
do sleep 2 ;
done" &
gnumeric
```

28.10.4 Mettere assieme le varie fasi di controllo

L'esempio seguente mostra le istruzioni necessarie a mettere assieme i vari controlli già descritti nelle sezioni precedenti, facendo riferimento al programma AbiWord:

```
if [ -x /usr/bin/abiword ]
then
    if ps x | grep -v grep | grep "[0-9] abiword" 2>&1 > /dev/null
    then
        xterm -e "echo Already running abiword ; sleep 10"
    else
        xterm -e "echo Loading abiword... ; sleep 2 ;
while ps rx | grep -v grep | grep \"[0-9] abiword\" ;
do sleep 2 ; done" & abiword
    fi
else
    xterm -e "echo Sorry, abiword is not available. ; sleep 15"
fi
```

Come già visto, con '**xmessage**' si può limitare l'uso di '**xterm**':

```
if [ -x /usr/bin/abiword ]
then
    if ps x | grep -v grep | grep "[0-9] abiword" 2>&1 > /dev/null
    then
        xmessage -timeout 10 "Already running abiword"
    else
        xterm -e "echo Loading abiword... ; sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] abiword\" ; ↩
↪do sleep 2 ; done" & abiword
    fi
else
    xmessage -timeout 15 "Sorry, abiword is not available."
fi
```

28.10.5 Innesto di un file system

«

Per controllare il comando '**mount**' si potrebbe usare un codice simile a quello seguente, in modo da sapere se l'operazione ha avuto successo o meno:

```
if mount /mnt/fd0
then
    xterm -e "echo /mnt/fd0  mounted successfully ; sleep 15"
elif mount | grep " /mnt/fd0 "
then
    xterm -e "echo /mnt/fd0  is already mounted ; sleep 15"
else
    xterm -e "echo Sorry: /mnt/fd0  mount failed ; sleep 15"
fi
```

L'esempio si basa sulla presenza di un file '`/etc/fstab`' nel quale è previsto il punto di innesto '`/mnt/fd0/`', che presumibilmente è riferito al file di dispositivo '`/dev/fd0`'.

28.10.6 Separazione di un file system

Per controllare il comando ‘**umount**’ si potrebbe usare un codice simile a quello seguente, in modo da sapere se l’operazione è ammissibile e se ha avuto successo:

```
if mount | grep " /mnt/hdc "
then
    if umount /mnt/hdc
    then
        eject /dev/hdc
        xterm -e "echo /mnt/hdc successfully unmounted ; sleep 15"
    else
        xterm -e "echo Sorry: umount failed ; sleep 15"
    fi
else
    xterm -e "echo /mnt/hdc is not to be unmounted. ; sleep 10"
fi
```

L’esempio presume che il file di dispositivo ‘`/dev/hdc`’ vada innestato nella directory ‘`/mnt/hdc/`’; inoltre, si suppone che si tratti proprio di un’unità che può espellere il supporto di memorizzazione (come nel caso di un lettore CD o DVD).

28.11 X: login grafico

X può essere avviato automaticamente, attraverso un sistema di autenticazione grafico, noto come *display manager*. In generale si tratta di un demone che viene configurato in modo da utilizzare una o più stazioni grafiche, locali o remote. Naturalmente, la configurazione predefinita di un sistema del genere, dovrebbe riguardare esclusivamente una sola stazione grafica locale.

È bene sottolineare che, quando si tratta di una sola stazione grafica locale, non c'è alcun bisogno di un sistema del genere e il buon vecchio script '**`startx`**' rimane la cosa migliore per avviare X, evitando in tal modo la presenza di un demone inutile nell'elenco dei processi elaborativi.

28.11.1 Configurazione generale

Nel momento in cui ci si inserisce un sistema grafico per l'autenticazione, prima dell'avvio di X, cambiano le dipendenze che ci sono tra i file di configurazione (script o porzioni di script). Pertanto, una configurazione personalizzata attraverso la modifica del file '`~/.xinitrc`', può rivelarsi inutile.

L'impostazione scelta da questa o quella distribuzione GNU può essere diversa e bisogna vedere i file di configurazione del sistema di autenticazione per sapere cosa succede veramente. Alcune distribuzioni GNU usano in particolare gli script '`/etc/X11/Xsession`' e '`~/.Xsession`' (o '`~/.xsession`'), rispettivamente per la configurazione globale e quella personalizzata di ogni utente. In questo modo, un utente che prima inseriva nel file '`~/.xinitrc`' le istruzioni per l'avvio del proprio gestore di finestre preferito, deve usare invece il file '`~/.Xsession`' per questo, ma nello stesso modo di prima.

Diversamente, in mancanza della configurazione corretta per l'avvio del gestore di finestre o di altro sistema del genere, dopo la fase di autenticazione, si avvia il solito servente X con un terminale e nulla altro.

```
#!/bin/sh
xsetroot -solid gray
xclock -digital -geometry +0-0 &
xbiff -geometry -0-0 &
exec fvwm
```

Quello che si vede sopra è il contenuto ipotetico di un file '`~/.Xsession`' predisposto per l'avvio del gestore di finestre Fvwm.

28.11.2 Utilizzo attraverso la rete

Il programma tipico che offre le funzionalità di un *display manager*, è in grado di fornire anche un accesso attraverso la rete, con il protocollo XDMCP (*X display manager control protocol*), che utilizza la porta 177. Quello che segue è un estratto dal file '`/etc/services`':

```
…
xdmcp           177/tcp    # X Display Manager Control Protocol
xdmcp           177/udp
…
```

In pratica, un elaboratore remoto deve avviare un proprio servente grafico, con il quale si collega al programma presso l'elaboratore che offre il servizio XDMCP. Il programma richiede l'autenticazione e da quel momento inizia una sessione grafica che riguarda l'elaboratore che offre il servizio, anche se viene gestita materialmente dallo schermo dell'elaboratore remoto.

Per collegarsi a un servizio XDMCP, ammesso che accetti effettivamente richieste di questo tipo, si può provare presso un elaboratore da usare come terminale un comando come quello seguente:

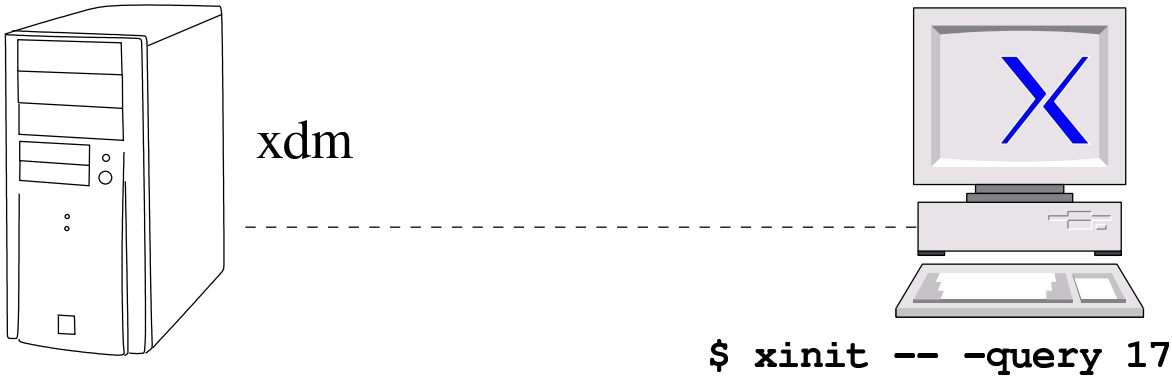
```
$ xinit -- -query 172.21.77.253 [Invio]
```

Oppure, direttamente così:

```
$ /usr/bin/X11/X vt7 -dpi 100 -query 172.21.77.253 [Invio]
```

In tal caso, l'indirizzo IPv4 172.21.77.253 è proprio quello dell'elaboratore che offre il servizio XDMCP.

Figura 28.112. Esempio di utilizzo di un servizio XDMCP.




```
$ xinit -- -query 172.21.77.253
```

oppure:

```
$ X -query 172.22.77.253
```

28.11.3 Avvio

«

Trattandosi di un sistema di autenticazione, il programma che se ne occupa potrebbe essere avviato tramite un record appropriato nel file '`/etc/inittab`', ma in generale si preferisce l'avvio di un demone attraverso la procedura di inizializzazione del sistema.

28.11.4 Xdm

«

Xdm[17] è il sistema di autenticazione grafica tradizionale di X, incluso anche in X.Org. In condizioni normali, i file e gli script di configurazione che lo riguardano, dovrebbero trovarsi nella directory '`/etc/X11/xdm/`'.

La configurazione predefinita dovrebbe prevedere l'apertura di una sola sessione grafica locale, escludendo l'accesso remoto.

L'avvio del demone '**xdm**', che si occupa di controllare l'accesso e l'avvio di X, dovrebbe essere gestito da uno script della procedura di inizializzazione del sistema; per esempio '`/etc/init.d/xdm`', o altro simile. Tuttavia, anche avviando direttamente l'eseguibile '**xdm**', si ottiene il risultato (naturalmente lo si deve fare con i privilegi dell'utente '**root**').

Una volta superata la fase di autenticazione in modo corretto, inizia una ***sessione***, controllata dallo script '`/etc/X11/xdm/Xsession`'. Al termine di questo script termina la sessione e ritorna la richiesta di autenticazione per quella stazione grafica. In condizioni normali, questo script dovrebbe richiamare un altro script usato anche per altri sistemi del genere (per esempio '`/etc/X11/Xsession`'), che a sua volta dovrebbe occuparsi di avviare lo script personalizzato dell'utente ('`~/.Xsession`' o '`~/.xsession`').

```
init-+
     |
    ...
     `-xdm-+-Xsession
           `-xdm
```

Quello che si vede sopra è l'interdipendenza tra i processi nel momento in cui il sistema di autenticazione attende che l'utente si presenti. Si può vedere che è necessaria la presenza del servente X e in particolare si può poi osservare che tutto funziona con i privilegi dell'utente '**root**'.

```
init-+
     |
    ...
     `-xdm-+-Xsession
           `-xdm---fvwm
```

Nel momento in cui si supera la fase dell'autenticazione, vengono avviati i processi richiesti dallo script '`/etc/X11/xdm/Xsession`' e dagli altri che questo richiama (per esempio '`~/.Xsession`'). In questo caso, come si vede nel riquadro precedente, si tratta del gestore di finestre Fvwm. Naturalmente, i processi avviati a partire dallo script '`/etc/X11/xdm/Xsession`' hanno i privilegi dell'utente che esegue l'autenticazione.

Il file di configurazione '`/etc/X11/xdm/xdm-config`' contiene l'elenco degli altri file utilizzati e di altre opzioni:

```
DisplayManager.errorLogFile:    /var/log/xdm.log
DisplayManager.pidFile:         /var/run/xdm.pid
DisplayManager.keyFile:         /usr/lib/X11/xdm/xdm-keys
DisplayManager.servers:         /usr/lib/X11/xdm/Xservers
DisplayManager.accessFile:      /usr/lib/X11/xdm/Xaccess
DisplayManager.authDir:         /var/lib/xdm
DisplayManager.willing:         su nobody -c /usr/lib/X11/xdm/Xwilling
! All displays should use authorization, but we cannot be sure X terminals
! will be configured to support it, so those that do not will require
! individual resource settings.
DisplayManager*authorize:       true
! Scripts to start the server, start the user session, and reset the server
DisplayManager*setup:           /usr/lib/X11/xdm/Xsetup
DisplayManager*startup:         /usr/lib/X11/xdm/Xstartup
DisplayManager*reset:           /usr/lib/X11/xdm/Xreset
DisplayManager*resources:       /usr/lib/X11/xdm/Xresources
DisplayManager*session:         /usr/lib/X11/xdm/Xsession
DisplayManager*authComplain:    true
DisplayManager*userPath:        /usr/local/bin:/usr/bin:/bin:↩
↪/usr/bin/X11:/usr/games
DisplayManager*systemPath:      /usr/local/sbin:/usr/local/bin:↩
↪/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11

! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

Nell'esempio merita molta attenzione l'ultima direttiva: la sua pre-

senza fa sì che venga a mancare il servizio XDMCP, pertanto i tentativi di collegamento verrebbero rifiutati. Per attivare il servizio, occorre commentare, o eliminare la direttiva:

```
…
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals
! with xdm
# DisplayManager.requestPort:   0
```

Secondo l'esempio mostrato, il file '`/etc/X11/xdm/Xservers`' permette di controllare l'avvio locale del servizio di autenticazione grafica. L'esempio seguente apre una sola sessione che deve collocarsi nella settima console virtuale:

```
…
:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp
```

Per avere due sessioni, una nella settima console, l'altra nell'ottava:

```
…
:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp
:1 local /usr/bin/X11/X vt8 -dpi 100 -nolisten tcp
```

Naturalmente, Xdm può servire anche solo per le connessioni remote, attraverso la rete, pertanto si può benissimo disattivare l'accesso locale, commentando le direttive di questo file.

Nella configurazione di Xdm c'è un altro file importante da considerare, che secondo la configurazione già vista è '`/etc/X11/xdm/Xaccess`'. Questo file serve a delimitare l'accessibilità del servizio XDMCP offerto. La direttiva dell'esempio seguente abilita l'accesso a qualunque indirizzo:

```
...
#
# Any host can get a login window:
#
*
...
```

28.11.5 Gdm

Gdm[18] è un altro sistema di autenticazione grafica conforme al gestore di sessione Gnome. Il principio di funzionamento è lo stesso di Xdm, dove in particolare è possibile scegliere di avviare sessioni differenti, che fanno riferimento a script diversi.

La configurazione e gli script di Gdm si trovano a partire dalla directory '`/etc/X11/gdm/`'; in particolare, gli script che consentono di selezionare delle sessioni diverse si trovano nella directory '`/etc/X11/gdm/Sessions/`'. Uno di questi dovrebbe fare riferimento allo script '`/etc/X11/Xsession`', il quale a sua volta si prende cura di avviare anche lo script personalizzato dell'utente ('`~/.Xsession`' o simile).

Una volta completata la fase di autenticazione, la dipendenza dei processi attivi potrebbe presentarsi in questo modo:

```
init-+
     |
    ...
     `-gdm---gdm-+-Xsession
                 `-fvwm2
```

Per la precisione, i processi avviati dagli script di sessione si trovano ad avere i privilegi dell'utente che si è autenticato, mentre il resto funziona con i privilegi dell'utente '**`root`**'.

Gdm è realizzato particolarmente per essere usato con il gestore di sessione Gnome; in tal caso, al posto di un gestore di finestre, viene avviato l'eseguibile '**gnome-session**', che a sua volta controlla un gestore di finestre, in base alla configurazione.

In condizioni particolari, Gdm si rifiuta di avviare la richiesta di autenticazione; ciò, in particolare, se manca la possibilità di verificare la presenza di un elaboratore corrispondente al nome restituito da '**hostname**'. Per esempio, l'indicazione corretta del nome, senza il dominio di appartenenza e senza una direttiva '**search**' appropriata nel file '`/etc/resolv.conf`', può provocare questo tipo di problema.

28.11.6 Kdm

Kdm[19] è un altro sistema simile a Xdm, con l'aggiunta della possibilità di selezionare delle sessioni differenti, come avviene per Gdm. La sua origine richiama il gestore di sessione KDE e i suoi file di configurazione potrebbero risiedere in '`/etc/kde2/kdm/`', oppure in '`/etc/X11/kdm/`'. A ogni modo, la struttura di questi file e di questi script è molto simile a quella usata da Xdm; anche in questo caso, il file '`kdm/Xsession`' dovrebbe rimandare allo script standard '`/etc/X11/Xsession`', il quale a sua volta dovrebbe utilizzare lo script personale degli utenti ('`~/.Xsession`' o '`~/.xsession`').

28.11.7 Wdm

«

Wdm[20] è un lavoro derivato da Xdm, i cui file di configurazione e gli script principali si collocano normalmente nella directory '`/etc/X11/wdm/`'. Anche in questo caso è disponibile la possibilità di avviare sessioni differenti.

28.12 Sessione

«

Nell'ambito di X, il termine «sessione» si inserisce nel momento in cui esiste un sistema di autenticazione che controlla l'utilizzo della stazione grafica, trattandosi infatti di una sessione di lavoro riferita a un certo utente.

In pratica, si tratta di un'astrazione ulteriore nel sistema di script che servono all'avvio del sistema grafico, dove '`/etc/X11/xinit/xinitrc`' e '`~/.xinitrc`' perdono il loro ruolo dominante, per passarlo allo script '`/etc/X11/Xsession`' e a '`~/.xsession`' o '`~/.Xsession`'.

28.12.1 Il problema che motiva il concetto di «sessione»

«

Le applicazioni che utilizzano il sistema grafico sono sempre più sofisticate e richiedono funzionalità che, da solo, X non può dare. Quindi, per poter usare certi programmi con certe funzionalità, si ha la necessità di avviarne altri, che eventualmente possono non mostrarsi, ma che comunicano con i primi.

Di norma, questi programmi che coadiuvano le attività svolte con il sistema grafico, non possono essere avviati prima che il servente X sia disponibile, inoltre devono operare con i privilegi dell'utente che ha avviato la sessione.

In un certo senso, è come se la sessione grafica corrispondesse all'avvio di un sottosistema personale, che richiede l'attivazione di certi servizi.

Dal momento che i servizi da avviare dipendono anche dai programmi che si intendono usare, diventa complicata e onerosa la compilazione di uno script '`/etc/X11/xinit/xinitrc`' che provveda a tutte queste cose, nel modo giusto. Pertanto, in un sistema operativo che prevede la grafica e anche le sessioni, lo script '**`xinitrc`**' si limita ad avviare a sua volta lo script '**`Xsession`**', il quale però non è da modificare, in quanto a sua volta esegue ciò che trova, in sequenza, nella directory '`/etc/X11/Xsession.d/`'; ed è lì, eventualmente, che si deve intervenire.

28.12.2 Gli script che controllano l'avvio della sessione

In un sistema che preveda le sessioni grafiche, lo script '`/etc/X11/xinit/xinitrc`' si limita a eseguire il contenuto dello script '**`Xsession`**', attraverso l'incorporazione:

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

# /etc/X11/xinit/xinitrc
#
# global xinitrc file, used by all X sessions started by xinit (startx)

# invoke global X session script
. /etc/X11/Xsession
```

In pratica, in questo modo il lavoro di '`/etc/X11/Xsession`' viene eseguito formalmente dallo stesso '**`xinitrc`**'.

Il codice contenuto nel file '`Xsession`' incorpora a sua volta, quello dei file contenuti nella directory '`/etc/X11/Xsession.d/`', se-

condo l'ordine lessicografico. Per esempio, tale directory potrebbe contenere i file seguenti:

```
$ ls /etc/X11/Xsession.d[Invio]

  /etc/X11/Xsession.d/20x11-common_process-args
  /etc/X11/Xsession.d/30x11-common_xresources
  /etc/X11/Xsession.d/50x11-common_determine-startup
  /etc/X11/Xsession.d/90x11-common_ssh-agent
  /etc/X11/Xsession.d/99x11-common_start
```

Al termine, se esiste il file '`~/.xsession`' (oppure '`~/.Xsession`'), viene eseguito anche il suo contenuto, altrimenti viene avviato ciò che si considera essere il «gestore di sessione» predefinito.

28.12.3 Gestori di sessione

«

Il gestore di sessione è un programma che dovrebbe facilitare l'uso del sistema grafico, ma che può anche complicarlo. Il suo compito è quello di consentire all'utente di configurare la propria sessione attraverso strumenti grafici, oltre che di facilitare la gestione dei servizi grafici necessari.

In pratica, se non si amano le complicazioni, si potrebbe usare benissimo un gestore di finestre puro e semplice, mentre un gestore di sessione vero e proprio si trova spesso a dover avvalersi a sua volta di un gestore di finestre.

Se si usa un sistema di autenticazione grafica sofisticato, può essere possibile la scelta tra diversi tipi di sessione, corrispondenti solitamente a script di avvio differenti, in cui può essere richiamato questo o quel gestore di sessione.

28.12.4 Gnome

Gnome [21] (pronunciato «g-a-n-o-m-e»), è un gestore di sessione altamente configurabile, contornato da diversi programmi realizzati specificatamente per rendere l'ambiente uniforme e confortevole. Naturalmente, i programmi realizzati per Gnome possono funzionare anche senza questo gestore di sessione: è sufficiente che siano disponibili le librerie grafiche necessarie.

Il programma eseguibile che rappresenta in pratica il gestore di sessione è '**gnome-session**'. Il fatto di avviarlo non produce nulla di particolare sullo schermo grafico, salvo che è possibile configurare l'avvio automatico di altri programmi di contorno, come per esempio il gestore di finestre.

A proposito del gestore di finestre, è da annotare che questo deve essere compatibile con Gnome; inoltre, se la propria distribuzione GNU non organizza bene i pacchetti, le prime volte che si avvia il gestore di sessione Gnome potrebbe essere complicato far partire anche il gestore di finestre. In mancanza d'altro, basta avviarlo da un terminale grafico, eventualmente dopo averlo ottenuto da un menù.

Infine, a proposito della dipendenza dei processi elaborativi, è da osservare che le applicazioni avviate dal gestore di sessione Gnome non risultano discendere da '**gnome-session**', ma direttamente dal processo principale (Init). Nell'esempio seguente si vedono diversi processi relativi a Gnome, staccati da '**gnome-session**', avviato a sua volta da Gdm:

```
init-+
     |
    ...
     |-enlightenment
     |-gdm---gdm-+-Xorg
     |           `-gnome-session---ssh-agent
     |-gnome-name-serv
     |-gnome-panel-pro
     |-gnome-smproxy
     |-gnome-terminal-+-bash
     |                `-gnome-pty-helpe
     |-gnomecc
    ...
     `-panel
```

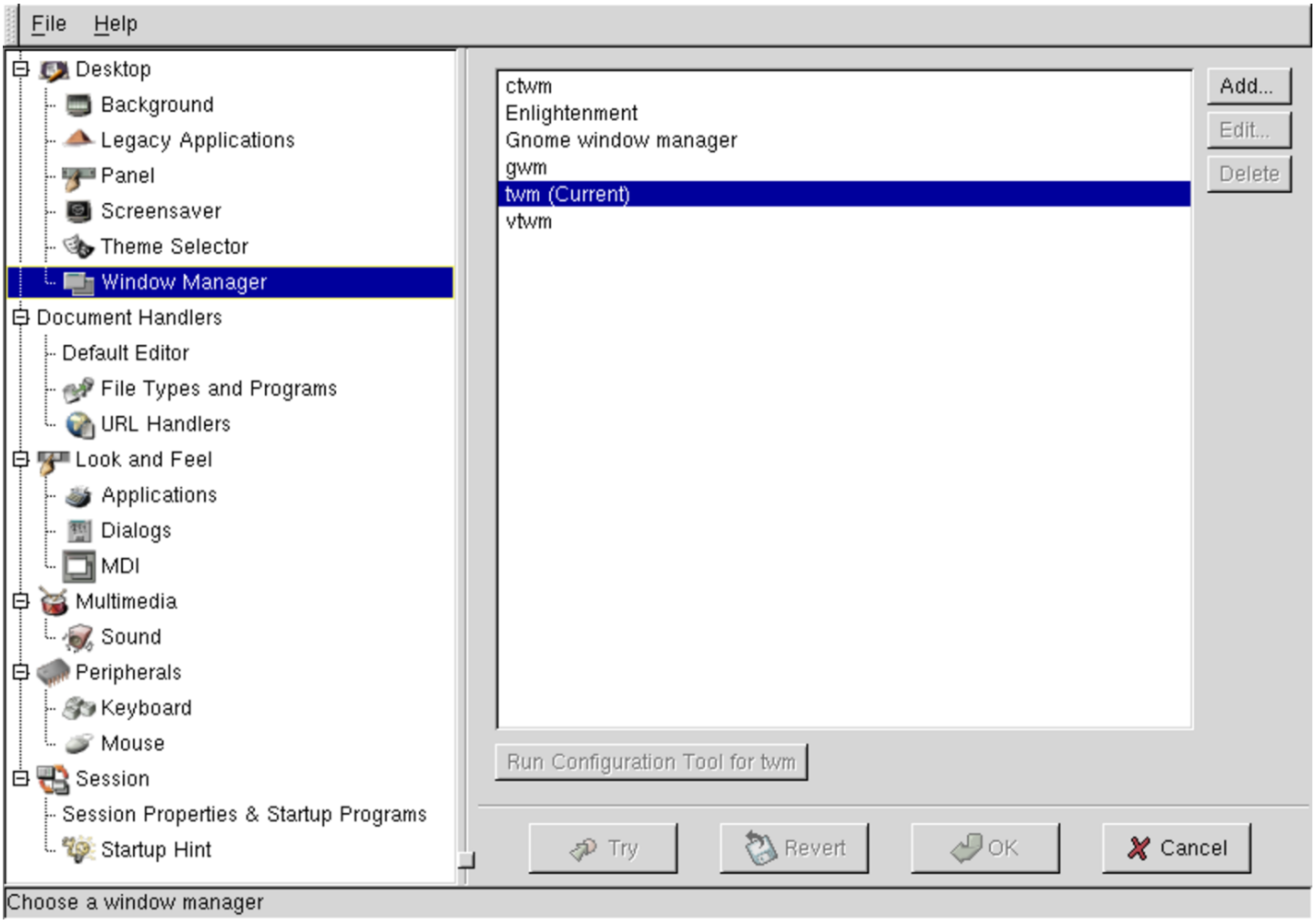
Nelle sezioni seguenti vengono descritti brevemente alcuni programmi essenziali per l'uso del gestore di sessione Gnome.

28.12.4.1 Gnome control center

«

Gnome control center è il programma di configurazione di Gnome. Lo si può avviare attraverso l'eseguibile '**gnomecc**', oppure da un menù, attraverso una voce che fa riferimento alle proprietà di qualcosa.

Figura 28.124. Gnome control center nella scelta del tipo di gestore di finestre.



28.12.4.2 Gnome panel

Gnome panel è un'altra applicazione importante che consente di avere sempre a portata di mano un menù da cui avviare le applicazioni. Sulla barra di questo menù possono risiedere anche delle icone per l'avvio rapido dei programmi più importanti, oppure delle ***applet*** (applicazioncine), ovvero programmi che graficamente hanno dimensioni ridotte. Il programma in questione corrisponde all'eseguibile '**`panel`**'.

Figura 28.125. Gnome panel, dove si vedono in particolare due bottoni per l'avvio rapido di applicazioni importanti.

28.12.5 KDE

«

KDE,[22] ovvero *K desktop environment*, è un altro gestore di sessione, paragonabile a Gnome. Il programma eseguibile che rappresenta in pratica il gestore di sessione è '**kde2**'. Come nel caso di Gnome, il fatto di avviarlo non produce nulla di particolare sullo schermo grafico, salvo il fatto che è possibile configurare l'avvio automatico di altri programmi di contorno, come per esempio il gestore di finestre.

A proposito della dipendenza dei processi elaborativi, è da osservare che le applicazioni avviate dal gestore di sessione KDE non risultano discendere da '**kde2**', ma da '**kdeinit**'; oppure, a seconda dei casi, possono discendere direttamente da Init. Nell'esempio seguente si vedono tre terminali aperti con la shell Bash:

```
init-+
     |
    ...
     |-kdeinit-+-artsd---artsd
     |         |-kdeinit
     |         `-3*[kdeinit---bash]
     |-7*[kdeinit]
     |-kdeinit---cat
     `-kdm-+-Xsession
           `-kdm---kde2-+-ksmserver
                        `-ssh-agent
```

È probabile che non si riesca a vedere se non si usa semplicemente '**ps**' per conoscere l'elenco dei processi attivi, ma esiste un program-

ma fondamentale per KDE, denominato '**kdesktop**', il cui scopo è quello di controllare il fondale, sul quale si depositano alcune icone. In altre parole, si tratta di un processo grafico che non si inserisce in una finestra e ricopre la superficie grafica, rappresentando in pratica la scrivania grafica di KDE.

In condizioni normali, KDE utilizza un menù alla base della superficie grafica dello schermo, corrispondente al programma '**kicker**', come si vede qui sotto:

I programmi che fanno parte della raccolta di KDE sono molti (spesso si tratta di derivazioni di altri programmi già esistenti, modificati in modo da usare le stesse librerie grafiche e armonizzare così l'estetica generale), caratterizzati comunemente dall'iniziale comune: «k». Alcuni di questi programmi sono molto importanti per l'ambiente e vengono descritti brevemente nelle sezioni seguenti.

28.12.5.1 Kpersonalizer

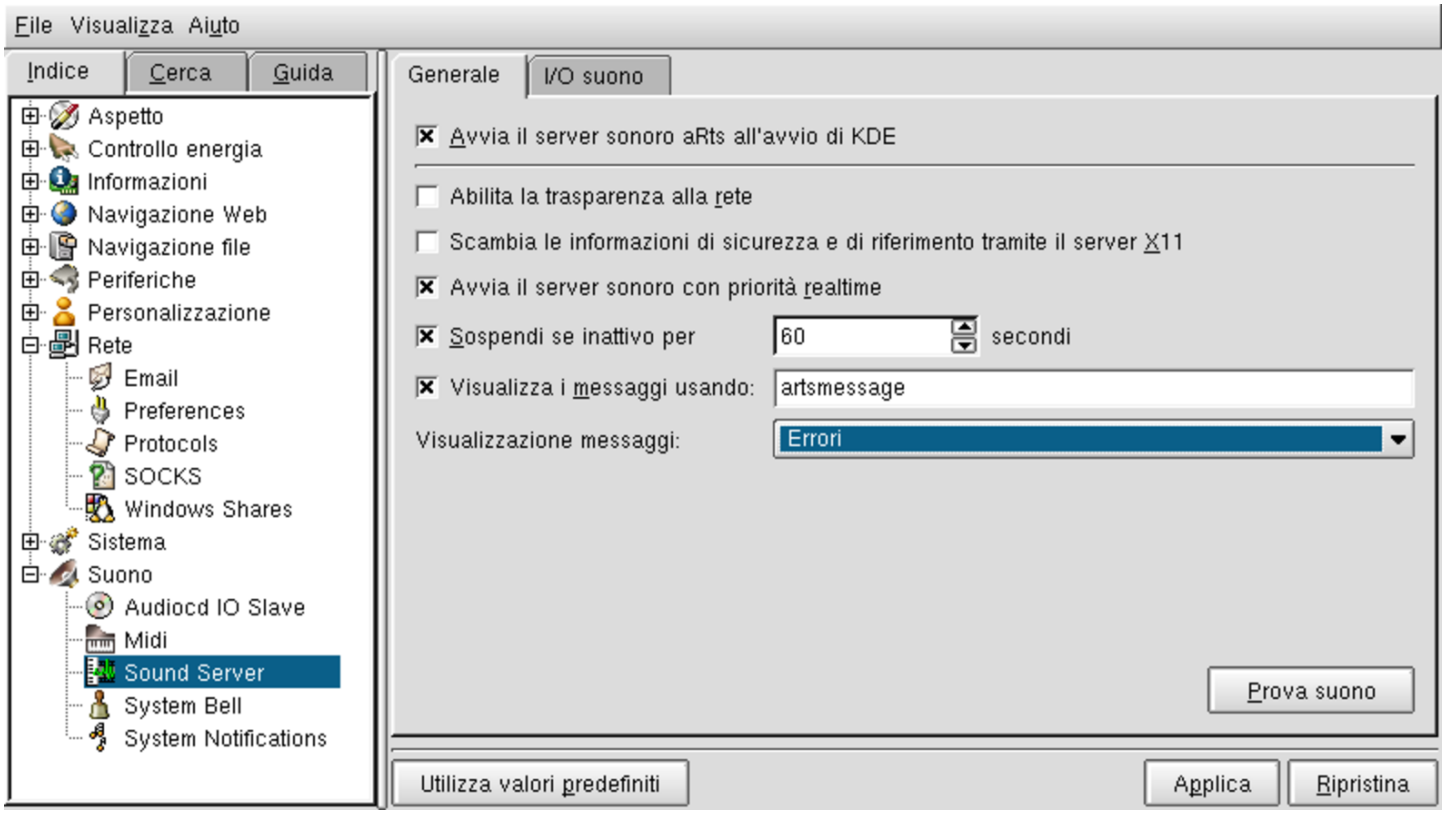
Kpersonalizer, a cui corrisponde l'eseguibile omonimo ('**kpersonalizer**'), consente una configurazione generale guidata. In particolare, come si vede nella figura 28.128, si imposta la disposizione della tastiera e la nazionalità.

Figura 28.128. Kpersonalizer.

28.12.5.2 Kcontrol

Lo stesso Kpersonalizer suggerisce di usare Kcontrol, ovvero il centro di controllo, alla fine della personalizzazione generale, per la definizione più dettagliata della configurazione di KDE. L'eseguibile corrispondente è '**`kcontrol`**'.

Figura 28.129. Kcontrol.

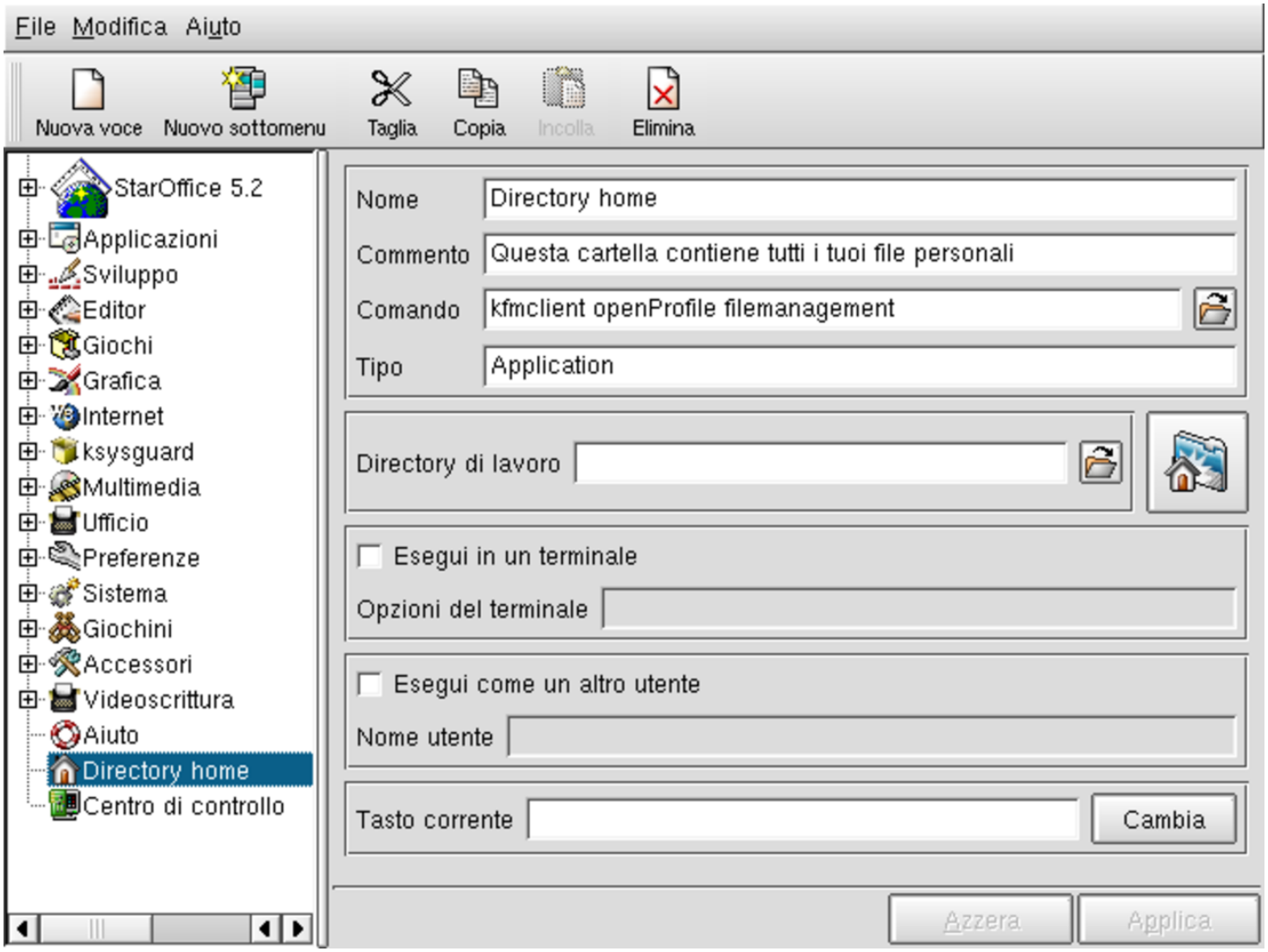


Kcontrol consente di visualizzare molti aspetti del sistema, anche se per questo, con GNU/Linux è sufficiente accedere alla directory '`/proc/`'.

28.12.5.3 Kmenuedit

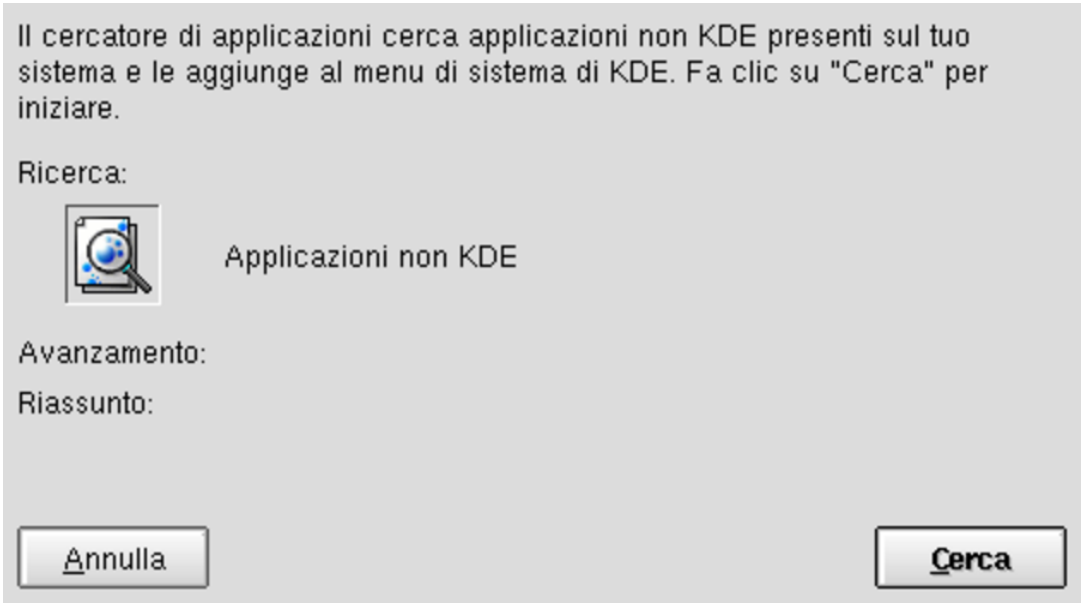
Kmenuedit consente di modificare agevolmente l'elenco delle voci contenute nel menù (il menù è gestito in pratica da '**`kicker`**').

Figura 28.130. Kmenuedit.



Eventualmente, è disponibile anche Kappfinder per ottenere una scansione automatica degli applicativi disponibili e il conseguente aggiornamento del menù in modo automatico.

Figura 28.131. Kappfinder.



28.12.5.4 Khelpcenter

Khelpcenter è un sistema integrato per la navigazione della documentazione. In pratica, consente di leggere la documentazione specifica di KDE, ma anche quella comune dei sistemi Unix e dei sistemi GNU (Info). La figura 28.132 mostra in particolare l'accesso alle pagine di manuale.

Figura 28.132. Khelpcenter.

28.13 Accesso remoto alla sessione di lavoro

«

Con un terminale a caratteri è possibile gestire una sessione di lavoro trasferibile successivamente in un altro terminale, attraverso il programma Screen (sezione 14.13); in modo simile, è possibile agire per quanto riguarda la sessione di lavoro con un servente X, attraverso VNC,[23] ovvero *Virtual network computing*.

È bene ricordare che X offre già la possibilità di eseguire un programma in un elaboratore, mostrandone le finestre nel servente di un altro (sezione 28.5.5). Diversamente da questa modalità comune di utilizzo di X, VNC consente di controllare tutto il servente grafico e il gestore di finestre (o anche il gestore di sessione) relativo. Inoltre, VNC consente anche un accesso simultaneo da parte di più

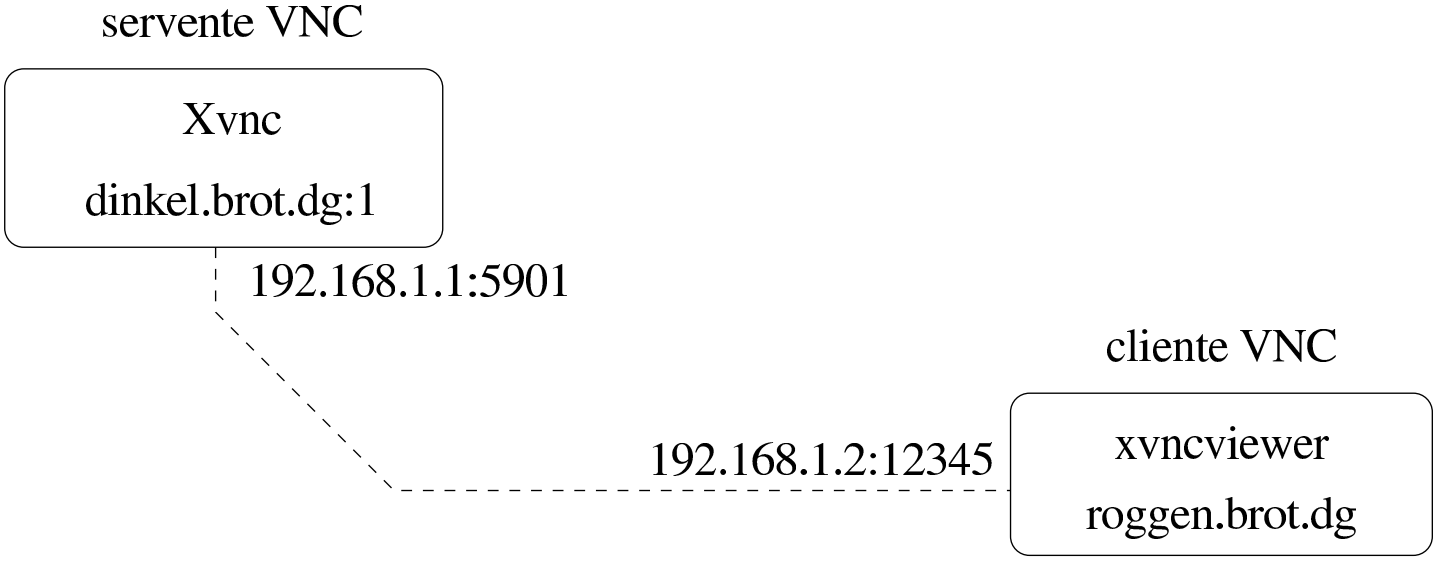
terminali remoti, solitamente per permettere la visualizzazione di ciò che avviene.

VNC è uno strumento utile soprattutto nell'ambito di una rete locale protetta dall'esterno, dal momento che utilizza una comunicazione in chiaro e che l'accesso al servente è controllato semplicemente da una parola d'ordine.

28.13.1 Funzionamento di VNC in generale

VNC si compone essenzialmente di un programma servente, con le funzionalità grafiche di X, il quale non si collega a una stazione grafica e viene usato attraverso un programma cliente apposito. Questo servente comunica nel modo consueto, come un servente X normale (connessioni TCP alle porte 6000+***n***) a cui si aggiunge la comunicazione necessaria al controllo attraverso il proprio cliente specifico, con le porte 5900+***n***.

Figura 28.133. Comunicazione tra il servente e il cliente VNC.



La figura mostra una situazione comune, in cui un elaboratore ospita un servente VNC, *dinkel.brot.dg*, che offre la stazione grafica virtuale '`:1`'. In tal modo, la comunicazione con il servente avviene alla porta 5901 (ovvero 5900 più il numero corrispondente alla stazione grafica virtuale).

Nell'elaboratore che ospita il servente VNC, l'interazione con questo non risulta apparente, a meno di avviare nello stesso un cliente VNC.

Il cliente VNC, a sua volta, potrebbe essere autonomo, oppure richiedere un servente X normale per poter funzionare. Il programma mostrato in figura è un esempio di cliente che richiede X per poter funzionare.

Un servente VNC può essere utilizzato da un solo cliente, oppure può essere consentito un accesso simultaneo da parte di più clienti; in tal caso, probabilmente, viene concesso a uno solo di interagire, mentre agli altri è permesso solo osservare. Pertanto, le situazioni più comuni di utilizzo di un sistema grafico basato su VNC sono due: l'esigenza di mantenere in funzione una sessione di lavoro grafica, a cui poter accedere da un terminale remoto, sospendendo e riprendendo la connessione anche da altre posizioni; oppure l'esigenza di fare un lavoro che altri utenti possono visualizzare, senza bisogno di un proiettore.

L'accesso a un servente VNC è controllato esclusivamente attraverso il confronto di una parola d'ordine, definita in modo indipendente dal meccanismo di riconoscimento degli utenti del sistema operativo.

28.13.2 Avvio e conclusione del funzionamento del servente VNC in un sistema GNU

Il funzionamento del servente VNC dipende dalla configurazione del servente X: se X non funziona correttamente a causa di un difetto di configurazione, anche il servente VNC non può funzionare. Pertanto, di solito si avvia un servente VNC da una sessione X già attiva, probabilmente da una finestra di terminale:

```
vncserver [:n_stazione_grafica] [opzioni]
```

Il programma '**vncserver**' è in realtà un involucro per controllare l'avvio di '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**' (dipende dall'edizione e ci possono essere anche altri nomi), che è invece il vero servente VNC.

Generalmente, l'avvio del servente VNC avviene sulla stazione grafica '**:1**', anche quando la stazione grafica '**:0**' non risulta impegnata, salvo che sia indicato diversamente con le opzioni della riga di comando.

Quando un utente avvia per la prima volta un servente VNC nel modo descritto, questo crea la directory '`~/.vnc/`', in cui vengono annotate le informazioni sulle sessioni di lavoro relative, oltre a un file contenente una parola d'ordine cifrata, che serve per consentire l'accesso successivo. In ogni caso, la prima volta provvede '**vncserver**' a preparare tutto; l'esempio seguente si riferisce all'utente '**tizio**' presso l'elaboratore *dinkel.brot.dg*:

```
$ vncserver[Invio]

  You will require a password to access your desktops.
```

```
Password: digitazione_all'oscuro [Invio]

Verify: digitazione_all'oscuro [Invio]

  New 'X' desktop is dinkel:1

  Starting applications specified in /etc/X11/Xsession
  Log file is /home/tizio/.vnc/dinkel:1.log
```

Come si può intendere, viene richiesta l'indicazione e la conferma di una parola d'ordine, che non può essere troppo breve, la quale viene conservata nel file '`~/.vnc/passwd`' in forma cifrata. Quando l'utente dovesse avviare nuovamente un servente VNC, disponendo già di questo file, non verrebbe più chiesta la parola d'ordine, rimanendo la stessa già stabilita in precedenza.

Superata questa fase, viene avviato effettivamente il servente VNC. Nell'esempio risulta avviato sulla stazione grafica virtuale '**`:1`**'; pertanto, per poterlo raggiungere, si deve usare un indirizzo del tipo *dinkel.brot.dg:1* (in generale conviene evitare la forma abbreviata che viene suggerita da '**`vncserver`**').

Al termine, '**`vncserver`**' ricorda dove si può trovare il file in cui sono annotate le informazioni specifiche sull'avvio del servente VNC, che diventa molto utile quando questo non si avvia come si desidera, per scoprire l'origine del problema. In generale, questo file ha la forma: '`~/.vnc/`***nodo*`:`***n_stazione_grafica*`.log`'.

Non sono da escludere problemi di configurazione di XFree86, che XFree86 stesso è in grado di superare, mentre il servente VNC non può.

Inizialmente, il contenuto di questo file può essere simile al testo seguente:

```
15/12/02 09:09:00 Xvnc version 3.3.5 - built Nov 22 2002 09:33:52
15/12/02 09:09:00 Copyright (C) 2002 RealVNC Ltd.
15/12/02 09:09:00 Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
15/12/02 09:09:00 All Rights Reserved.
15/12/02 09:09:00 See http://www.realvnc.com for information on VNC
15/12/02 09:09:00 Desktop name 'X' (dinkel:1)
15/12/02 09:09:00 Protocol version supported 3.3
15/12/02 09:09:00 Listening for VNC connections on TCP port 5901
```

Eventualmente, se sono stati installati i componenti necessari di VNC, '**vncserver**' avvia il servente VNC in modo da offrire anche un accesso HTTP alla porta 5800+***n***, per mezzo del quale, con un navigatore in grado di mettere in funzione programmi Java, è possibile accedere in mancanza di un programma cliente migliore. In tal caso, si può osservare questo fatto nello stesso file appena mostrato:

```
15/12/02 09:09:00 Listening for HTTP connections on TCP port 5801
15/12/02 09:09:00   URL http://dinkel:5801
```

In questo caso, l'indirizzo per accedere è preferibilmente *http://dinkel.brot.dg:5801*.

Se si vuole avviare il servente VNC senza avviare prima X, le cose si complicano un po'. Infatti, quando ciò è possibile, X determina da solo alcune informazioni sul funzionamento dell'adattatore grafico e sulle capacità dello schermo reale; pertanto, quando si avvia il servente VNC da una sessione di X già attiva, le stesse informazioni vengono utilizzate da VNC, mentre in mancanza di queste, il funzionamento di VNC dipenderebbe da parametri predefiniti, spesso non gradevoli. Per esempio, avviando un servente VNC senza l'appoggio di un servente XFree86 preesistente, si ottiene una stazione grafica impostata sostanzialmente per un adattatore di tipo VGA

standard (640×480 punti grafici e una profondità di colori modesta). L'esempio seguente mostra l'avvio di un servente VNC, attraverso '**vncserver**', al di fuori di una sessione di lavoro con X, dove si specifica la dimensione della superficie grafica (1024×768 punti grafici) e la profondità di colori (16 bit):

```
$ vncserver -depth 16 -geometry 1024x768 [Invio]
```

Per concludere il funzionamento del servente VNC, presso l'elaboratore locale, si può usare '**vncserver**' con l'opzione '**-kill**':

```
vncserver -kill :n_stazione_grafica
```

Al termine della descrizione dell'avvio di un servente VNC, è bene chiarire che, quando accede un cliente VNC, se già esiste un altro programma cliente collegato, generalmente questo (quello preesistente) termina di funzionare. In pratica, in condizioni normali, si suppone che l'utente che accede a un servente VNC sia l'unico autorizzato a farlo, pertanto, se è rimasta una sessione aperta, ciò è dovuto probabilmente a una dimenticanza dello stesso. Per consentire degli accessi simultanei al servente VNC, è necessaria l'opzione '**-alwaysshared**', come descritto nella tabella seguente, che riepiloga alcune opzioni per l'avvio dell'involucro '**vncserver**'.

Tabella 28.138. Alcune opzioni della riga di comando di '**vncserver**'.

Opzione o argomento	Descrizione
:*n_stazione_grafica*	Consente di indicare espressamente il numero di stazione grafica da utilizzare. In mancanza di questa informazione dovrebbe trattarsi di '**:1**'.
`-geometry` ***n_punti_larghezza*`x`*n_punti_altezza***	Definisce la dimensione della superficie grafica da utilizzare, in punti grafici.
`-depth` ***n_bit***	Definisce la profondità di colori espressa in numero di bit.

`-nevershared`
 	
`-alwaysshared`
	Consente di impedire o consentire la condivisione multipla del servente. In pratica, la prima delle due opzioni è quella predefinita, per fare in modo che solo un accesso per volta sia permesso.

`-startup` ***programma***	Avvia all'interno del servente grafico il programma o lo script indicato.
`-kill` **:*n_stazione_grafica***	Conclude il funzionamento del servente, in funzione nell'elaboratore locale, controllando la stazione grafica indicata.

28.13.3 Avvio del servente VNC in condizioni difficili in un sistema GNU

«

L'impostazione effettiva di un servente X in una distribuzione GNU, può essere molto complessa. In altri termini, il funzionamento di '**xinit**' e di '**startx**', non è perfettamente uniforme da una distribuzione all'altra, spesso per la necessità di arginare dei problemi di sicurezza. Pertanto, qualsiasi sia la ragione, può succedere che un servente VNC non si comporti come ci si aspetterebbe; si può arrivare anche a vedere funzionare il servente, ma senza un gestore di finestre o un gestore di sessione.

Di fronte a problemi di questo tipo, può essere più conveniente avviare direttamente il servente VNC senza l'aiuto dell'involucro '**vncserver**', predisponendo uno script adatto alle proprie esigenze. Vengono mostrati qui due script: uno per controllare '**Xvnc**', ovvero l'eseguibile del servente VNC, l'altro per controllare l'avvio di un gestore di finestre, chiamato all'interno del primo. A fianco di questi esempi, ne vengono mostrati comunque anche di equivalenti in cui si utilizza '**vncserver**', ma in modo da ottenere lo stesso risultato, perché alle volte può essere vero il contrario, ovvero che senza '**vncserver**' non si riesca ad avviare VNC.

Come già accennato, il programma eseguibile del servente VNC può essere denominato '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**', a seconda dell'edizione. Tuttavia, è normale che sia disponibile almeno un collegamento simbolico che consenta l'uso del nome '**Xvnc**'.

```
#!/bin/sh
# vncs1024

# X fonts.
VNC_FONTS=\
/usr/lib/X11/fonts/misc,\
/usr/lib/X11/fonts/75dpi,\
/usr/lib/X11/fonts/100dpi/,\
/usr/lib/X11/fonts/Type1/,\
/usr/lib/X11/fonts/Speedo/

# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
vncpasswd ~/.vnc/passwd

# Start VNC server at screen :1, using ~/.vnc/log for
# log file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -dpi 100 2> ~/.vnc/log &

# Start the window manager inside a wrap script.
# After window manager run, the VNC server should be killed,
# by the wrap script.
vncwm &
```

```
#!/bin/sh
# vncwm
xloadimage -display :1 -onroot -fullscreen ~/.wallpaper
fvwm -display :1  2> /dev/null ; killall Xvnc ; \
```

```
                                    killall Xrealvnc ; \
                                    killall Xtightvnc
```

Il primo di questi due script, denominato qui '**vncs1024**', definisce inizialmente una variabile di ambiente, contenente l'elenco delle directory dei caratteri di X (questa informazione può essere tratta eventualmente dal file di configurazione di X: '`/etc/X11/xorg.conf`'). Successivamente elimina i processi avviati con il nome '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**', ammesso che ce ne siano, poi elimina anche il contenuto della directory '`~/.vnc/`'; quindi chiede di definire una parola d'ordine nuova, con l'aiuto di '**vncpasswd**'.

Quando tutto è pronto, si avvia l'eseguibile '**Xvnc**', utilizzando la stazione grafica '**:1**' (come fa normalmente '**vncserver**'), con un elenco piuttosto lungo di opzioni, come si può vedere dall'esempio. In particolare, in questo caso, si specifica una dimensione della superficie grafica di 1024×768 punti grafici, inviando il flusso dello standard error nel file '`~/.vnc/log`', per poter sapere ciò che accade. Si osservi inoltre che '**Xvnc**' viene avviato sullo sfondo in modo esplicito.

Infine si avvia uno script '**vncwm**', il cui scopo è quello di avviare un gestore di finestre e di chiudere il funzionamento di '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**', al termine del funzionamento di questo. Infatti, come si vede, lo script carica un fondale e avvia Fvwm: al termine del funzionamento di Fvwm elimina tutti i processi con il nome '**Xvnc**', '**Xrealvnc**' e '**Xtightvnc**'.

Naturalmente, in questo modo si può avviare un solo servente VNC alla volta.

Da quanto visto si intuisce la sintassi per l'avvio dell'eseguibile '**Xvnc**':

```
Xvnc [:n_stazione_grafica] [opzioni]
```

Segue la descrizione di alcune opzioni della riga di comando.

Tabella 28.141. Alcune opzioni della riga di comando di '**Xvnc**'. Si osservi che a seconda della realizzazione di VNC, alcune opzioni potrebbero non funzionare.

Opzione o argomento	Descrizione
:n_stazione_grafica	Consente di indicare espressamente il numero di stazione grafica da utilizzare. In mancanza di questa informazione si tenta di usare '**:0**'.
`-auth` *file*	Definisce il nome del file usato per l'autenticazione di X.
`-geometry` *n_punti_larghezza*`x`*n_punti_altezza*	Definisce la dimensione della superficie grafica da utilizzare, in punti grafici.
`-depth` *n_bit*	Definisce la profondità di colori espressa in numero di bit.

Opzione o argomento	Descrizione
`-nevershared` `-alwaysshared`	Consente di impedire o consentire la condivisione multipla del servente. In pratica, la prima delle due opzioni è quella predefinita, per fare in modo che solo un accesso per volta sia permesso.
`-rfbwait` ***n_millisecondi***	Definisce il tempo massimo di attesa, in millisecondi, per un cliente VNC.
`-rfbauth` ***file***	Definisce il file contenente la parola d'ordine cifrata che serve per consentire l'accesso.
`-rfbport` ***n_porta***	Definisce il numero della porta usata per la comunicazione con i clienti VNC.
`-fp` ***stringa***	Definisce l'elenco dei percorsi delle directory contenenti informazioni sui caratteri tipografici da usare.

Opzione o argomento	Descrizione
`-co` ***file***	Definisce il file contenente le informazioni sui colori, senza l'estensione '`.txt`'. Generalmente si ricopia il percorso indicato nella direttiva '**`RgbPath`**' del file '`/etc/X11/xorg.conf`'.
`-dpi` ***n_punti***	Definisce la risoluzione in punti per pollice.

In alternativa all'avvio diretto di '**`Xvnc`**', il comando di avvio potrebbe essere sostituito così:

```
…
# Start VNC server at screen :1, using ~/.vnc/log for log
# file.
vncserver :1 -auth ~/.Xauthority -geometry 1024x768 \
    -depth 16 -rfbwait 120000 -rfbauth ~/.vnc/passwd \
    -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -dpi 100 \
    -startup true 2> ~/.vnc/log &
…
```

Si può osservare la comparsa dell'opzione '**`-startup`**', con la quale si vuole evitare che '**`vncserver`**' avvii autonomamente un gestore di sessione o altro, che comunque viene già controllato all'interno del proprio script.

28.13.4 Configurazione e utilizzo dei caratteri tipografici

Non esiste una configurazione vera e propria del servente VNC; esiste piuttosto una configurazione di '**vncserver**', che di solito si lascia commentata completamente. In ogni caso, si tratta del file '`/etc/vnc.conf`' ed eventualmente di '`~/.vncrc`'.

L'utilizzo di questi file diventa utile, quindi, solo se si avvia il servente VNC attraverso '**vncserver**' e si sono manifestati dei problemi a cui si pone rimedio solo con la configurazione.

Una situazione in cui è necessario intervenire nella configurazione è la presenza di direttive '**FontPath**' nel file di configurazione di X ('`/etc/X11/xorg.conf`'), che fanno riferimento a caratteri tipografici non esistenti; per esempio quando queste informazioni sono fornite da un servente di caratteri che in quel momento non risulta rispondere. In tal caso, si può specificare nella configurazione quali percorsi sono sicuri, tralasciando il superfluo.

28.13.5 Accesso a un servente VNC

Si può accedere a un servente VNC con diversi programmi, ma in un sistema GNU dovrebbe essere preferibile farlo attraverso '**xvncviewer**'. Come lascia intuire il nome, si tratta di un programma che richiede l'uso di un servente X già attivo, che mostra poi la stazione grafica remota in una finestra di quella locale.

Di solito è necessario avviare '**xvncviewer**' da una finestra di terminale, per poter specificare a quale nodo di rete e a quale stazione grafica collegarsi. Si utilizza la sintassi seguente:

```
xvncviewer [opzioni] nodo:n_stazione_grafica
```

Se nella riga di comando non viene specificata l'opzione '**-passwd**' (con la quale si indica un file contenente una parola d'ordine cifrata), è necessario inserire la parola d'ordine per l'accesso al servente VNC:

```
$ xvncviewer dinkel.brot.dg:1 [Invio]

  VNC viewer version 3.3.5 - built Nov 22 2002 09:31:25
  Copyright (C) 2002 RealVNC Ltd.
  Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
  See http://www.realvnc.com for information on VNC.
  VNC server supports protocol version 3.3 (viewer 3.3)

Password: digitazione_all'oscuro [Invio]
```

Successivamente vengono visualizzate altre informazioni, quindi appare la finestra relativa alla comunicazione con il servente VNC.

Se quello che si vede è solo uno sfondo grigio, senza applicazioni attive, dove premendo i tasti del mouse non si ottiene nulla, è probabile che il servente VNC sia stato avviato senza un gestore di finestre o un gestore di sessione.

Quando si vuole visualizzare semplicemente ciò che accade in un servente VNC, senza poter interferire, mentre un altro cliente VNC sta interagendo, si può usare l'opzione '**-viewonly**', assieme a '**-shared**'. Eventualmente, dalla parte del servente si può usare l'opzione '**-alwaysshared**' per garantire che sia consentito l'accesso simultaneo da parte di più clienti (questa opzione vale sia per l'avvio diretto di '**Xvnc**', sia per l'involucro '**vncserver**'):

```
$ xvncviewer -viewonly dinkel.brot.dg:1 [Invio]
```

Segue la descrizione di alcune opzioni.

Tabella 28.144. Alcune opzioni della riga di comando di '**xvncviewer**'.

Opzione o argomento	Descrizione
nodo*:*n_stazione_grafica	Consente di indicare il nodo di rete e il numero di stazione grafica da utilizzare.
`-shared`	Richiede una connessione a un servente VNC, consentendo esplicitamente la condivisione dello stesso con altri clienti.
`-viewonly`	Richiede una connessione a un servente VNC per la sola visualizzazione di ciò che accade. Di solito si usa assieme all'opzione '**-shared**'.
`-fullscreen`	Fa in modo di funzionare occupando tutta la superficie disponibile, senza il contorno di una finestra.
`-passwd` ***file***	Consente di specificare un file contenente una parola d'ordine cifrata, che dovrebbe corrispondere a quanto utilizzato dallo stesso servente, in modo da non richiedere all'utente l'inserimento della stessa.

28.13.6 Utilizzo comune di VNC

«

La situazione in cui è più comune l'utilizzo di VNC è quella dell'utente che si trova lontano dal proprio elaboratore, al quale può comunque accedere attraverso la rete. In generale, in questo elaboratore remoto non è già in funzione alcun servente VNC, pertanto conviene avviare X nell'elaboratore di cui si dispone temporaneamente; quindi, da lì, con una finestra di terminale, si può contattare l'elaboratore remoto e avviare il servente VNC. Se tutto funzio-

na correttamente, il servente VNC viene avviato con caratteristiche compatibili alla grafica di cui si dispone effettivamente; quindi ci si può collegare con un cliente VNC.

elaboratore_locale`$ ssh tizio@dinkel.brot.dg` [*Invio*]

In questo modo ci si collega all'elaboratore *dinkel.brot.dg* utilizzando l'utenza '**`tizio`**'. Successivamente si avvia il servente VNC presso l'elaboratore remoto:

elaboratore_remoto`$ vncserver` [*Invio*]

Quindi, se tutto ha funzionato correttamente ci si collega con un cliente VNC:

elaboratore_remoto`$ exit` [*Invio*]

elaboratore_locale`$ xvncviewer dinkel.brot.dg:1` [*Invio*]

28.13.7 VNC attraverso un tunnel cifrato con il protocollo SSH

Attraverso Secure Shell (sezione 44.7) è possibile creare un tunnel cifrato, per utilizzare con più tranquillità l'accesso a un servente VNC. Viene riproposto l'esempio di utilizzo comune, utilizzando un tunnel del genere:

elaboratore_locale`$ ssh tizio@dinkel.brot.dg` [*Invio*]

In questo modo ci si collega all'elaboratore *dinkel.brot.dg* utilizzando l'utenza '**`tizio`**'. Successivamente si avvia il servente VNC presso l'elaboratore remoto:

elaboratore_remoto`$ vncserver` [*Invio*]

Dall'elaboratore locale ci si collega nuovamente con l'elaboratore remoto per creare un tunnel cifrato:

elaboratore_remoto$ **exit** [*Invio*]

elaboratore_locale$ **ssh -N -L 5901:dinkel.brot.dg:5901** [*Invio*]

A questo punto, invece di contattare direttamente l'elaboratore remoto *dinkel.brot.dg*, è invece sufficiente collegarsi a quello locale; prima però, conviene mettere il programma '**ssh**' sullo sfondo:

[*Ctrl z*]

elaboratore_locale$ **bg** [*Invio*]

elaboratore_locale$ **xvncviewer localhost:1** [*Invio*]

28.13.8 Inserire VNC automaticamente all'avvio di X

«

È possibile realizzare uno script con cui si avvia un servente VNC e subito dopo X con un cliente VNC, a tutto schermo, che punta esattamente al servente locale, senza interferire con l'utente. Questo tipo di tecnica può servire in un laboratorio didattico in due casi: quando l'insegnante vuole avviare una sessione di lavoro grafica, pronta subito perché gli studenti vi si possano collegare, evitando così di utilizzare un proiettore; quando si vuole fare avviare agli studenti la sessione di lavoro grafica in modo che l'insegnante abbia la possibilità di intervenire sul loro lavoro, senza doversi spostare fisicamente dalla sua postazione.

```
#!/bin/sh
# vncsc1024

# X fonts.
VNC_FONTS=\
```

```
/usr/lib/X11/fonts/misc,\
/usr/lib/X11/fonts/75dpi,\
/usr/lib/X11/fonts/100dpi/,\
/usr/lib/X11/fonts/Type1/,\
/usr/lib/X11/fonts/Speedo/

# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
vncpasswd ~/.vnc/passwd

# Start VNC server at screen :1, using ~/.vnc/log for log file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -alwaysshared nologo -dpi 100 2> ~/.vnc/log &

# Start the window manager inside a wrap script.
# After window manager run, the VNC server should be killed, by
# the wrap script.
vncwm &

# Start xinit with xvncviewer as a client
xinit /usr/bin/xvncviewer -fullscreen -passwd ~/.vnc/passwd localhost:1
```

Come si può osservare, questo esempio è molto simile a quanto già visto in una sezione precedente, dove la novità sta nell'avviare, dopo lo script '**vncwm**', '**xinit**' specificando l'avvio di '**xvncviewer**' al posto del solito gestore di finestre. Naturalmente, lo script '**vncwm**' rimane tale e quale a prima:

```
#!/bin/sh
# vncwm
xloadimage -display :1 -onroot -fullscreen ~/.wallpaper
fvwm -display :1  2> /dev/null ; killall Xvnc ; \
```

```
                                    killall Xrealvnc ; \
                                    killall Xtightvnc
```

Come si può intuire, lo script che qui è stato chiamato '**vncsc1024**' è adatto per l'insegnante (o il relatore) che vuole consentire l'accesso ai suoi studenti, a cui deve comunicare anche la parola d'ordine per accedere, che come si vede viene sostituita ogni volta. Diversamente, se si vuole realizzare uno script da fare usare agli studenti al posto del solito '**startx**', si deve fare in modo che il file della parola d'ordine sia già stato preparato e sia «standard»; l'estratto seguente mostra solo le istruzioni salienti da modificare:

```
# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
cp /etc/vnc/sharedx.passwd ~/.vnc/passwd
chmod 0600 ~/.vnc/passwd
```

Anche qui è possibile utilizzare '**vncserver**' con l'aggiunta dell'opzione '**-startup**':

```
…
# Start VNC server at screen :1, using ~/.vnc/log for log
# file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -alwaysshared nologo -dpi 100 \
    -startup true 2> ~/.vnc/log &
…
```

28.13.9 RealVNC e TightVNC

I due filoni principali dello sviluppo di VNC sono rappresentati da RealVNC e da TightVNC. In generale, questi lavori sono abbastanza compatibili tra di loro, comunque vale la pena di conoscere i nomi con cui potrebbero essere distinti i programmi e gli script che li riguardano:

VNC	VNC4	RealVNC	TightVNC
‘`Xvnc`’	‘`Xvnc4`’	‘`Xrealvnc`’	‘`Xtightvnc`’
‘`vncserver`’	‘`vnc4server`’	‘`realvncserver`’	‘`tightvncserver`’
‘`xvncviewer`’	‘`xvnc4viewer`’	‘`xrealvncviewer`’	‘`xtightvncviewer`’
‘`vncconnect`’	‘`vnc4connect`’	‘`realvncconnect`’	‘`tightvncconnect`’
‘`vncpasswd`’	‘`vnc4passwd`’	‘`vncpasswd.real`’	--

28.13.10 Script «vncrc»

Il file *allegati/vncrc* è uno script, da prendere come esempio da adattare e migliorare, con lo scopo di facilitare l’uso di VNC nelle situazioni più comuni.

28.13.11 Conclusione

VNC è un lavoro che ha prodotto diversi filoni di sviluppo, per esigenze differenti. Oltre a quanto descritto in questo capitolo esistono diversi altri programmi che possono essere di un certo interesse. Di tale disponibilità è bene tenerne conto, per sapere che può essere utile una ricerca approfondita prima di organizzare il proprio lavoro in modo sistematico con VNC.

28.14 Composizione video

«

La «composizione video» è l'arte con cui si elaborano le immagini di diverse fonti video per produrre quegli effetti che si usano per esempio in televisione. Può trattarsi di rotazioni, deformazioni, effetti tridimensionali, trasparenze e simili.

X non offre potenzialità del genere e per ottenerle occorre quello che è noto come *composition manager*, ovvero un «gestore di composizione», il quale si occupa di sfruttare eventuali funzionalità hardware già presenti nell'interfaccia grafica, per questo scopo.

Nelle situazioni più comuni, le funzionalità di un gestore di composizione sono incorporate nel gestore di finestre, chiamandosi in tal caso *compositing window manager*.

Da un punto di vista operativo, si può usare benissimo un sistema grafico privo di funzionalità per la composizione video; tuttavia, ci sono programmi che hanno la necessità di avvalersi di queste: per esempio, il programma Ardesia (*http://code.google.com/p/ardesia/*) non può funzionare senza.

Se non si utilizza un gestore di finestre che integra già le funzionalità di composizione video e l'unico scopo è quello di poter usare software che non può farne a meno, la soluzione più semplice e meno invasiva, è rappresentata probabilmente da Xcompmgr. Lo si avvia così, in uno degli script che controllano la sessione grafica:

```
...
xcompmgr &
...
```

L'avvio del gestore di finestre deve avvenire successivamente.

28.15 Il futuro di X

La gestione della grafica attraverso X è diventata ormai una questione fin troppo complessa, tanto che esistono diversi tentativi di proporre piattaforme alternative, più semplici e più snelle. Allo stato attuale (2011) l'alternativa più promettente è rappresentata da Wayland (*http://wayland.freedesktop.org*), di cui è bene tenere conto.

28.16 Riferimenti

- Wikipedia, *X Window System*, *http://en.wikipedia.org/wiki/X_Window_System* , *http://it.wikipedia.org/wiki/Compositing_window_manager*
- Wikipedia, *Compositing window manager*, *http://en.wikipedia.org/wiki/Compositing_window_manager* , *http://it.wikipedia.org/wiki/Compositing_window_manager*
- *X.Org foundation*, *http://www.x.org/*
- *The XFree86 Project, Inc.*, *http://www.xfree86.org/*
- Eric S. Raymond, *XFree86 Video Timings HOWTO*, *http://www.linuxdoc.org/HOWTO/XFree86-Video-Timings-HOWTO/index.html*
- *ISO/IEC 9995:1994, Information technology -- Keyboard layouts for text and office systems*, *http://www.iso.org*
- *Pictogrammes ISO 9995-7*, *http://pages.infinit.net/pm2/lexique4.htm*
- Erik Fortune, *The X keyboard extension: protocol specification*, *http://www.xfree86.org/current/XKBproto.pdf*

- Kamil Toman, Ivan U. Pascal, *The XKB configuration guide*, http://www.xfree86.org/current/XKB-Config.pdf
- Kamil Toman, Ivan U. Pascal, *How to further enhance XKB configuration*, http://www.xfree86.org/current/XKB-Enhancing.pdf
- Doug Palmer, *An unreliable guide to XKB configuration*, http://www.charvolant.org/~doug/xkb/
- Ivan U. Pascal, *X keyboard extension*, http://pascal.tsu.ru/en/xkb/
- Matt Chapman, *Window managers for X*, http://www.plig.org/xwinman/ .
- *FVWM*, http://www.fvwm.org
- *RealVNC*, http://www.realvnc.com
- *TightVNC*, http://www.tightvnc.org
- Karl Runge, *x11vnc: a VNC server for real X displays*, http://www.karlrunge.com/x11vn11c/
- Giuseppe De Marco, *Cripting delle sessioni VNC*, articolo contenuto nella rivista *Linux magazine*, edizioni Master, ISSN 1592-8152, anno III, numero 25, dicembre 2002
- Wikipedia, *Wayland (display server protocol)*, http://en.wikipedia.org/wiki/Wayland_(display_server_protocol)

[1] **X.Org** MIT più altre licenze per porzioni particolari di codice

[2] **XFree86** MIT più altre licenze per porzioni particolari di codice

[3] **X** MIT più altre licenze per porzioni particolari di codice

[4] **X** MIT più altre licenze per porzioni particolari di codice

[5] **Xfstt** GNU LGPL

[6] Se si vuole provare a vedere cos'è un servente X senza clienti basta avviare '**X**'. Come già spiegato in precedenza, è sempre possibile uscire con la combinazione [*Ctrl Alt Backspace*].

[7] In questo caso, dal momento che '**fvwm**' viene avviato rimpiazzando la shell, risulta che il processo di '**xclock**' dipende proprio da '**fvwm**'.

[8] Prima di utilizzare '**xon**' è indispensabile sapere gestire '**rsh**'.

[9] I caratteri tipografici di X servono solo per la rappresentazione di testo sullo schermo. In pratica, non sono utili per la stampa vera e propria.

[10] **X** MIT più altre licenze per porzioni particolari di codice

[11] **X** MIT più altre licenze per porzioni particolari di codice

[12] **X** MIT più altre licenze per porzioni particolari di codice

[13] **X** MIT più altre licenze per porzioni particolari di codice

[14] **X** MIT più altre licenze per porzioni particolari di codice

[15] **SCIM** GNU GPL

[16] **Fvwm 2** software libero soggetto a diverse licenze a seconda della porzione di codice coinvolta

[17] **X** MIT più altre licenze per porzioni particolari di codice

[18] **Gdm** GNU GPL

[19] **KDE** GNU GPL, GNU LGPL e altre licenze a seconda della porzione di codice

[20] **Wdm** GNU GPL con l'aggiunta della licenza specifica di Xdm

[21] **Gnome** GNU GPL e GNU LGPL

[22] **KDE** GNU GPL, GNU LGPL e altre licenze a seconda della porzione di codice

[23] **VNC** GNU GPL