eXperimentando, o qué cosas se pueden hacer con las X

Javier Quintano
Instituto Saturnino de la Peña
javier@jaxvinet.homelinux.org
versión 0.2
15 de noviembre de 2004

El sistema gráfixo ’X’ consiste en algo más que un entorno en el que ejecutar aplicaciones con interface gráfica, su arquitectura de red, nos permite ejecutar aplicaciones gráficas en una máquina de forma remota, o usar la potencia gráfica de esa máquina en nuestro terminal local. Este documento da pistas sobre el uso del servidor X para estas labores.

Introducción

Para un usuario ajeno que se acerca a Linux por primera vez, le resulta transparente el modo de funcionamiento de el sistema gráfico del sistema operativo. Así podríamos decir que las aplicaciones que necesitan "dibujar" algo en pantalla, manejarse con una ventana etc... se ejecutan sobre otro programa común a ellas: el servidor X. Pero además como ocurre con otras muchas aplicaciones este programa tiene una arquitectura de red, que hace posible que varias aplicaciones lo usen como clientes al mismo tiempo, o incluso que distintos usuarios, o máquinas lo utilizen. A ésto podemos añadirle el detalle de que ese uso puede hacerse desde el mismo ordenador donde se ejecuta el servidos de X o también desde un ordenador remoto.

Hoy por hoy esta filosofía, hace que el sitema gráfico sea algo más lento que en otros SO en los que el entorno gráfico está totalmente integrado con el resto del sistema operativo, lo cual es un origen de cuelgues constantes por fallos en la programación de las aplicaciones que lo usan o del propio SO. Por otro lado tiene la ventaja de ser más flexible y potente, así por ejemplo imagina hacer cosas como estas:

- Tener varios gestores de ventanas funcionando a la vez en el mismo PC (kde,gnome,wmaker...)
- Tener un gestor de ventana distinto iniciado para el usuario root, en la tecla Ctrl+Alt+F8
- Permitir que un usuario(distinto del que está usando el entorno gráfico en ese momento) de nuestro PC inicie desde su línea de comando una aplicación gráfica sobre nuestra pantalla.
- Hacer un ssh a un ordenador e iniciar remotamente un KDE en la máquina remota, para después ejecutar aplicaciones de forma también remota sobre ese KDE.
- Hacer un ssh a la máquina remota y ejecutar los xeyes sobre el KDE del usuario que está arrancado en ese momento en ese PC.
- Hacer un ssh a la máquina remota y ejecutar en nuestro portátil el gimp instalado en esa máquina.
- Logear directamente y de manera gráfica sobre un ordenador remoto, y usar todos sus recursos gráficos desde nuestra máquina local.
- Arrancar sobre una ventana de nuestro wm (window manager, wmaker, kde ..) otro wm distinto para probarlo, o arrancar una distro que tengamos instalada en un HD externo por usb o lo que sea que funcione sobre un display de X !!!

Soy consciente de la mezcla de términos e ideas que empiezan a aparecer ya a estas alturas, no he intentado evitarlo y poco a poco iremos afinando técnicamente las expresiones usadas para describir el cúmulo de cosas que podemos hacer con nuestras X ...

El sistema X Window

En principio decir que estamos haciendo un tratamiento genérico del tema, ya que existen distintas aplicaciones que funcionan en algunos gestores como KDE para hacer algunas de las taréas que auí describiré, pero me parece más interesante tratar el tema a más "bajo nivel" que va a ser útil para entender mejor el funcionamiento real del sistema. Por otro lado serán muy poco las líneas de comando que emplearemos, y muy poco los archivos de configuración editados, ya que una vez que entendemos el fundamento del servidor X, es muy fácil enredar con las distintas cosas que podemos hacer.

Como usuarios de Linux, conocemos y usamos habitualmente el entorno de texto (consola) pero sabemos que hay entre ese modo y nuestro KDE ? Cuántas aplicaciones, servicios, comunicaciones entre procesos tienen lugar para que nuestro entorno X funcione? Qué ocurre cuando movemos el ratón sobre una ventana, cuanzo hacemos click con el botón o movemos el scroll de la rueda central, cuando desplazamos una ventana ...? Prueba a ejecutar:

```
jaxvi@portable:~$ xev
```

Como era de esperar, hay un montón de eventos, hilos etc ... funcionando. Empecemos por el principio, qué son "las X" ?

El Sistema X Window, conocido tambien como X, es una colección de programas que forman una interface para el usuario, orientada al uso de teclado, ratón (mouse) y una pantalla gráfica.Este software ha sido desarrollado por el MIT (Massachusetts Institute of Technology), y debido a su diseño independiente del hardware corre en una gran variedad de estaciones de trabajo (workstations) de diferentes proveedores.

El modelo cliente/servidor

X está contruido alrededor de un modelo Cliente/Servidor. Esto significa que ciertos programas X juegan los papeles de Cliente y otros programas, diferentes, juegan el papel de Servidor. El trabajo se logra por la comunicación entre ellos. El Cliente y el Servidor pueden estar en el mismo computador, o en diferentes computadores, tal como decíamos en la introducción. Ambos ordenadores se comunican como no con un protocolo conocido como: The X Window System Protocol (Protocolo del Sistema de Ventanas X).
NOTA: Para las pruebas descritas, hemos trabajado en una red local con dos ordenadores: el ordenador considerado como mi host local, mi portátil:"portable-192.168.0.5". La máquina remota:"torre-02-192.168.0.2".

Vamos a intentar una primera experiencia. Intentaremos ejecutar un xterm del ordenador remoto(torre-02), sobre unas X nuevas iniciadas en nuestro ordenador local(portable).

Iniciamos unas X en nuestro terminal 8, para lo que vamos a tty1, la primera consola, Ctrl+Alt+F1 y logeamos como root, hacemos:

```
portable:~# xinit -- :1.0
```

Nótese que se inician las X sin gestor de ventanas, tan sólo iniciando un xterm por defecto.

Iniciamos una conexión remota a la máquina torre-02:

```
Ctrl+Alt+F2 , login
portable:~# ssh 192.168.0.2
Password:
torre-02:~#
```

Ejecutamos un xterm, pero mandándolo a nuestro nuevo servidor X en el el display8:

```
torre-02:~# xterm -display 192.168.0.5:1.0
Xlib: connection to "192.168.0.5:1.0" refused by server
```

Nos dice que el servidor no le deja hacer conexiones, para solucionarlo vamos a nuestro servidor nuevo, al xterm que nos ha iniciado y ponemos:

```
portable:~# xhost +192.168.0.2
192.168.0.2 being added to access control list
portable:~#
```

Volvemos a la conexión remota y ahora volvemos a poner:

```
torre-02:~# xterm -display 192.168.0.5:1.0
```

Si hacemos Ctrl+Alt+F8 y conectamos al servidor nuevo, veremos que se ha iniciado un xterm, que se ha superpuesto al antiguo que teníamos, y que pertenece a la máquina remota:

```
torre-02:~# uname -a
Linux torre-02 2.4.25-1-386 #2 Wed Apr 14 19:38:08 EST 2004 i686 GNU/Linux
```

OT:y si, este kernel es insecure total desde local XDD

Pues aunque no lo parezca, ya hemos manejado bastantes cosas de las X. Hemos iniciado un nuevo servidor X, y hemos dejado que se conecten a el remotamente, e inicien aplicaciones en el. Lo mismo podríamos haber hecho alho un poco más bruto, pero seguimos con la teoría y lo intentamos más adelante...

El Servidor

El servidor de X11 es el programa responsable de dibujar en la pantalla de la estación de trabajo. Cada aplicación envía comandos al servidor de X11 tales como : Coloca una ventana aquí, dibuja una línea de aquí para alla, colorea esta sección azul, etc. Cuando recibe estos comandos, el servidor de X11 desempeña las operaciones apropiadas para que la estación de trabajo o el terminal realice la acción o muestre la información adecuada.

La razón de que el Sistema X ponga toda la responsabilidad del despliege en pantalla en el servidor X11, es para hacer más fácil la escritura de programas de aplicación. Diferentes fabricantes usan diferentes métodos para controlar el despliegue en pantalla - Los comandos para dibujar una línea en una Estación de Trabajo Sun no trabajarían en una Estación DEC, etc . Los programas pueden enviar comandos simples como dibujar una línea al servidor y no preocuparse acerca de cómo se dibuja una línea en cada tipo de Estación de Trabajo.

Los Clientes

Los programas , llamados clientes de X11, son el corazón del sistema X Window, y la parte con que la que realmete trabajamos, recordemos en la intro el paso de consola a gráfico... Cada programa es llamado cliente porque requiere que esa acción sea procesada, para ello, por el servidor X11. Hay tres programas que entenderemos fácilmente:el manejador de ventanas(KDE,Gnome,WMAKER...), el emulador de terminal, y el reloj X, estas son las que vamos a usar para juguetear, pero cada aplicación gráfica que iniciemos se comunica de manera similar con el servidorX.

El concepto de display

A pesar de haber usado un nuevo servidor X, es muy posible que aún no sepamos que es un display, un monitor, una pantalla, un desktop etc ... y no, no son todo lo mismo ;-)

Un display es el conjunto de monitor(es), teclado y ratón que están bajo el control de un servidor X. Cada display puede tener más de un monitor y cada máquina puede tener más de un display. Cuando se ejecuta una aplicación X ésta necesita saber a qué servidor debe conectarse, esto se hace mediante una variable de entorno que contiene el host y el display a cuyo servidor X se va a conectar y el monitor dentro de ese display donde va a aparecer. El host es opcional, si se omite se entiende que se trata de un display local y el monitor también es opcional, si se omite se entiende el monitor número 0. Esta variable de entorno se llama DISPLAY:

```
jaxvi@portable:~$ echo $DISPLAY
:0.0
```

En mi caso la variable tiene el valor ":0", esto quiere decir que cualquier aplicación X que ejecute se conectará al servidor X que controla el display 0 local y se verá en el monitor 0 de ese display, este valor sería equivalente a:

```
127.0.0.1:0.0
```

En la parte del host se admiten tanto direcciones IP como nombres, siempre y cuando algo resuelva el nombre, esto puede ser cómodo a veces, o necesario en redes compartidas con máquinas win/samba.

```
jaxvi@portable:~$ cat /etc/hosts
127.0.0.1       localhost
192.168.0.5     portable
192.168.0.2     torre-02
```

Pues bien, recordemos que en la experiencia de arriba, para decirle a la *aplicación clienteX11* a qué servidor X queríamos mandarle le hemos pasador el parámetro *-display 192.168.0.5:1.0*, esto no habría sido necesario, si para esa sesión exportamos la variable de entorno, como sigue:

```
torre-02:~# export DISPLAY=192.168.0.5:1.0
```

Pues bien, retomémos la práctica. Vamos a hacer tal como prometimos algo un poquito más potente.Iniciamos, como ya sabemos un nuevo display en el terminal8, desde el tty1. Logeamos como usuario, el mismo que escribe (jaxvi) en el tty2. Voy a abrir en ese nuevo display mi gestor de ventanas favoritos, en este caso WindowMaker (wmaker)

```
jaxvi@portable:~$ wmaker
/usr/bin/WindowMaker fatal error: no se pudo abrir el display ""
jaxvi@portable:~$ export DISPLAY=:1
...
```

Ahora si!! Hacemos el mítico Ctrl+Alt+F8 y ahí está otro wmaker funcionando !Por cierto que ahora desde ahí podríamos probar algo así como:

```
jaxvi@portable:~$ echo $DISPLAY
:1.0
En una xterm del wmaker principal (el de F7 ) hacemos:
jaxvi@portable:~$ xclock -display :1
```

Por último volvemos al segundo wmaker(F8)y ahí está nuestro reloj!

Vamos a ir un poco más allá...

```
Ctrl+Alt+F1 y le damos Ctrl+C para matar las X del terminal8.
portable:~# startx -- :1
```

o incluso...

```
Matamos el terminal, y arrancamos de nuevo con
portable:~# xinit -- :1.0
```

en el xterm que se nos abre junto con esas X hacemos:

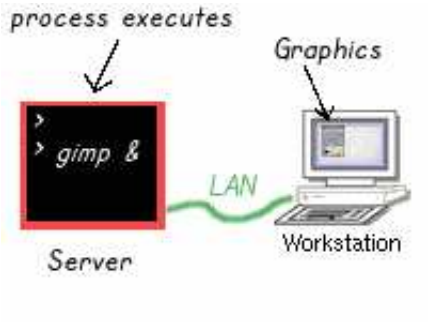
```
portable:~# startkde
```

La verdad es que las posibilidades y maneras son infinitas, sólo tenéis que jugar con la imaginación y/o vuestras necesidades.

Ahora veremos como pasar a lo siguiente no tiene ninguna complicación.

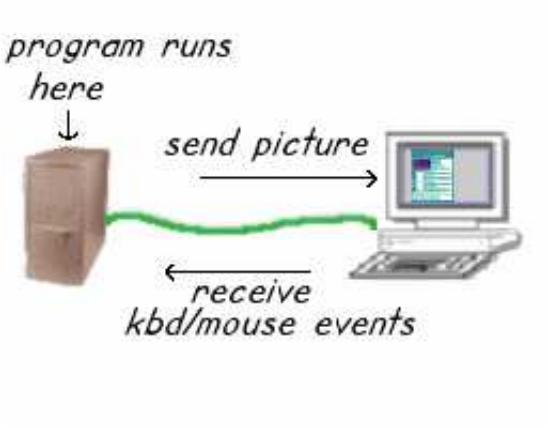
Ejecutando aplicaciones X remotas

A estas alturas ya debería estar resuelto, la manera de lanzar aplicaciones de manera remota sobre nuestro ordenador y viceversa, de hecho era nuestro primer ejemplo. Pero veamos porqué esto puede ser interesante. Como hemos visto,cada aplicación gráfica en X Window lee al iniciarse la variable de entorno DISPLAY para averiguar a qué pantalla debe enviar sus gráficos. Esto junto con la capacidad de red del Sistema X Window hace posible ejecutar aplicaciones gráficas remotamente. Es decir, se utiliza la capacidad de CPU de una máquina mientras se utiliza la aplicación desde otra máquina distinta. Todo el GUI (graphical user interface, interfaz gráfica de usuario) aparece en la máquina desde la que se opera. No se nota que se utilizan dos ordenadores. Por supuesto la velocidad de la red es importante, pero una conexión de LAN normal de 10Mbit/s es más que suficiente.



¿Por qué es esto interesante? Hay muchas aplicaciones de estas "redes gráficas". Las empresas las utilizan para operar remotamente con equipos que pueden estar a miles de kilómetros de distancia y pudiendo usar la misma aplicación para controlarlos que la que se utilizaría si se estuviese en el mismo lugar en el que se encuentran los equipos. Podríamos tener 2 ordenadores: una máquina rápida a 1GHZ y un viejo Pentium 133MHz. Podemos aprovechar la velocidad de la nueva máquina aunque no se estemos sentados frente a

ella. ¿Cómo funciona? Los clientes X(nuestros programas como gimp, netscape, algún programa que gestiona una máquina de producción ...) envían al servidor instrucciones sobre cómo pintar cuadros y botones. A cambio reciben del servidor los eventos de ratón y teclado. Obviamente se necesita algún tipo de autentificación, pero ya hemos visto cómo solucionar esto con el comando *xhost*. Una manera más avanzada sería usando *xauth*.Con xauth se puede dar acceso a los usuarios de forma individual. Es mucho más seguro que xhost. La autentificación usa una cookie alojada en el fichero .Xauthority en el directorio personal de los usuarios. Si la máquina remota tiene una cookie válida en este fichero entonces se garantizará el acceso.



Atención porque en estos momentos, se está hablando de servidor X refiriéndose al portátil, que es la máquina de menor potencia, sobre ella, sobre este servidor, se ejecutan los clientes de la máquina potente torre-02. No os preocupéis mucho por esto, es más una cuestión de costumbre que nos lleva a pensar en la máquina potente como la máquina servidor.

Cuando conectamos a una máquina remota usando su entorno gráfico, su KDE por ejemplo, el servidor X sigue estando en nuestra máquina local, aunque hablemos de esa máquina remota como "la que nos está sirviendo las X", ya que realmente estamos trabajando sobre su KDE...

Veremos eso en el siguiente apartado Terminal-X. Antes vamos a hacer un par de prácticas.Primero y aunque no sean realmente demasiado útiles, iniciaremos un nuevo display desde un ssh a la máquina remota sobre el que ejecutar alguna aplicación gráfica, tanto en el display iniciado, como en el display que se encontraba funcionando ya.

Después dejaremos claro con un ejemplo cómo iniciar aplicaciones remotamente que es lo que hemos descrito más arriba, y lo que nos interesa realmente.

Hacer un ssh a un ordenador e iniciar remotamente un KDE en la máquina remota, para después ejecutar aplicaciones de forma también remota sobre ese KDE:

```
torre-02:~# xhost +192.168.0.5
jaxvi@portable:~$ ssh root@192.168.0.2
torre-02:~# startx -- :1
ejecutado desde otra conexión:
torre-02:~# xeyes -display :1 (lo podríamos hacer exportando la variable de entorno)
```

Hacer un ssh a la máquina remota y ejecutar los xeyes sobre el KDE del usuario que está arrancado en ese momento en ese PC.

```
torre-02:~# xeyes -display :0
```

Iniciar aplicaciones remotamente:

```
En el terminal tty2
jaxvi@portable:~$ startx -- :1
En el nuevo display iniciamos un xterm y ejecutamos xhost +

En otro terminal
jaxvi@portable:~$ ssh root@192.168.0.2
torre-02:~# export DISPLAY=192.168.0.5:1.0
torre-02:~# konqueror & (lo mandamos a 2º plano para poder seguir usando el shell)
```

Volvemos a nuestro terminal 8 , y sobre nuestro gestor de ventanas, se estará ejecutando el konqueror de la máquina remota. Es curioso comprobar que el manejador de ventanas es una parte independiente de la aplicación, así si en vez de usar *startx* para iniciar el nuevo display hubiésemos hecho *xinit*, de modo que habríamos iniciado unas X, y al iniciar el konqueror, se iniciaría sín ventana.

He de decir aquí que el trabajar de esta manera se puede automatizar usando scripts o tocando archivos de configuración, exportando las variables para que sea más o menos transparente o al menos más cómodo para el usuario.

Una utilidad muy interesante es el forwarding de X de ssh. Esto significa que nos ahorramos andar exportando variables etc ... El propio cliente/servidor ssh se encarga de reenviar a nuestro servidor local lo que estemos ejecutando desde ssh y necesite de X.

Sería algo así:

```
portable:/home/jaxvi# ssh -X 192.168.0.2
Password:
Last login: Sun Nov 14 20:41:16 2004 from 192.168.0.5
```

Ahora si ejecutamos un xterm, se iniciará la aplicación de X remotas, en nuestras X. Para comprobarlo podemos ver el $DISPLAY desde esa sesión ssh:

```
torre-02:~# echo $DISPLAY
localhost:10.0
```

He tenido algún problema con el forwarding de las X, ya que no me permite hacerlo como usuario normal, por algún bloqueo de .Xauthority, pero ni siquiera como root. Dependiendo de la distribución que usemos y sus políticas de seguridad tendremos que revisar el siguiente archivo y editarlo si es necesario:

```
torre-02:~# less /etc/ssh/sshd_config
```

debemos poner el siguiente valor como sigue:

```
X11Forwarding yes
```

De cualquier manera si lo que se quiere es trabajar realmente contra las X de un ordenador remoto (x-terminal), veremos cómo hacerlo en el siguiente apartado.

X-Terminal - exportando las X

Como decía, hasta ahora hemos jugueteado con algunas de las cosas que podemos hacer con las X. Esto nos ha podido servir para entender su funcionamiento, pero a la hora de llevarlo a la práctica, podemos necesitar dar un paso más.Primerio unos previos teóricos.

Display managers

Para mostrar como se consigue que un ordenador se comporte como un X Terminal tenemos que conocer lo que son los display managers, el que viene con las XFree86 se llama xdm, GNOME tiene uno llamado gdm y KDE tiene otro llamado kdm.Recientemente han intentado ponerse de acuerdo respecto a cómo/dónde albergar los scripts de los distintos gestores de ventanas etc ... Pero en principio cada DM tiene su propio archivo de configuración aunque sean algo distintos. En clase trabajamos con KDM, así que veremos cómo configurar este gestor, para mas cachondeo sobre una suse :P .

Total, una vez que el display manager está ejecutándose es capaz de recibir peticiones de otros servidores X, que pueden estar en la misma máquina o en otras máquinas de la red, esto se hace mediante el protocolo XDMCP (X Display Manager Control Protocol). El display manager escucha en un puerto (por defecto es el 177) peticiones de distintos servidores X, cuando recibe una petición hará que en el servidor solicitante aparezca el prompt para que los usuarios hagan login, el login se producirá en la máquina que ejecuta el display manager, no en la que ejecuta el servidor, esta última tan sólo ejecuta el servidor X, gracias a esto se puede convertir casi cualquier máquina en un X Terminal y utilizar desde él un escritorio completo.

Configurando GDM

Para que el display manager acepte peticiones hay que configurarlo correctamente, basta con editar el fichero /etc/gdm/gdm.conf, en la sección llamada [xdmcp] dar valor true a la opción Enable, con esto basta, aunque hay otras opciones que se pueden configurar: el número máximo de sesiones, el puerto en el que escuchar, etc. Son opciones bastante sencillas y el manual de gdm lo explica bastante bien, además gdm viene con un programa de configuración bastante intuitivo llamado gdmconfig.

Así de sencillo, y esto no es sino un frontend de lo que podemos editar en el archivo de configuración, como decía: /etc/gdm/gdm.conf, la sección correspondiente sería:

```
# XDMCP is the protocol that allows remote login.  If you want to log into
# gdm remotely (I'd never turn this on on open network, use ssh for such
# remote usage that).  You can then run X with -query <thishost> to log in,
# or -indirect <thishost> to run a chooser.  Look for the 'Terminal' server
# type at the bottom of this config file.
[xdmcp]
# Distributions: Ship with this off.  It is never a safe thing to leave
# out on the net.  Setting up /etc/hosts.allow and /etc/hosts.deny to only
# allow local access is another alternative but not the safest.
# Firewalling port 177 is the safest if you wish to have xdmcp on.
# Read the manual for more notes on the security of XDMCP.
Enable=true
# Honour indirect queries, we run a chooser for these, and then redirect
# the user to the chosen host.  Otherwise we just log the user in locally.
#HonorIndirect=true
# Maximum pending requests
#MaxPending=4
#MaxPendingIndirect=4
# Maximum open XDMCP sessions at any point in time
#MaxSessions=16
```

```
# Maximum wait times
#MaxWait=15
#MaxWaitIndirect=15
# How many times can a person log in from a single host.  Usually better to
# keep low to fend off DoS attacks by running many logins from a single
# host.  This is now set at 2 since if the server crashes then gdm doesn't
# know for some time and wouldn't allow another session.
#DisplaysPerHost=2
# The number of seconds after which a non-responsive session is logged off.
# Better keep this low.
#PingIntervalSeconds=15
# The port.  177 is the standard port so better keep it that way
Port=177
```

Configurando KDM

Aunque parezca una bobada lo más difícil de configurar en Suse+kde, es encontrar el archivo de configuración sobre el que los cambios tienen efectos realmente.

En general para kdm deberíamos hacer lo siguiente:

Dar permisos al kdm en Xaccess, descomentar para que aparezca la siguiente línea:

```
joe /etc/kde3/kdm/Xaccess
*                                       #any host can get a login window
```

Dar permisos al kdm en kdmrc, comprobar que está enabled:

```
joe /etc/kde3/kdm/kdmrc

# Whether KDM should listen to incoming XDMCP requests.
# Default is true
Enable=true
# The UDP port KDM should listen on for XDMCP requests. Don't change.
# Default is 177
Port=177
# File with the private keys of X-terminals. Required for XDM authentication.
# Default is ""
#KeyFile=/etc/kde3/kdm/kdmkeys
```

Reiniciar kdm y comprobar que esta xdm:

```
/etc/init.d/kdm restart
ps -e | grep xdm
```

Por último en suse+kdm, tendremos que modificar lo siguiente:

```
joe /etc/X11/xdm/Xaccess

*    CHOOSER BROADCAST                          #any indirect host can get a chooser
```

```
joe /etc/opt/kde3/share/config/kdm/kdmrc

[xdmcp]
Enable=true
Port=177
```

Por supuesto reiniciar xdm o kdm ..

```
/etc/init.d/kdm restart
```

o

```
/etc/init.d/xdm restart
```

esto depende de nuestros script de inicio, y como no dispongo de una suse donde probar, si es necesario, lo revisaré en el futuro. :S

Ni que decir tiene que una distribución como Suse, hará virguerías con artificios como "exportr escritorio kde" etc ... que supongo que por detrás tendrán un vnc o así funcionando. La verdad, esas opciones nunca las trabajo, será un defecto personal ...

Por último recomiendo usar el sistema gráfico de configuración del kdm tal com vemos a continuación:

Conectando a un x-terminal

Pero, ¿ cómo aprovechamos esta característica de los display managers?, pues muy sencillo. Retomando la práctica vamos a tty1, y ponemos:

```
portable:/home/jaxvi# X -query 192.168.0.2 :1
```

Si vamos por partes, le estmos diciendo que queremos iniciar unas nuevas X, que las vamos a pedir a un host remoto, y que las vamos a iniciar sobre un nuevo display en nuestro servidorX local (que irá a tty8). Fácil no?, pero podemos hacer más cosas.

Peticiones de broadcast: se buscará en la red un host dispuesto a atender una petición. El comando es el siguiente:

```
portable:/home/jaxvi# X -broadcast :1
```

Peticiones indirectas: gdm pondrá en el display una aplicación llamada chooser, que permitirá al usuario elegir el host de la red al que quiere entrar. Se hace con este comando:

```
portable:/home/jaxvi# X -indirect 192.168.0.2 :1
```

Pero de nuevo esto se queda probe si pretendemos montar un aula con un servidor al que se conectan terminales tontos, o al menos no gráficos. Para esto desearemos que al arrancar la máquina ella sóla haga todo el trabajo, expandiendo un login del server. Para que en el arranque se abra el servidor X y se realize la petición XDMCP, es suficiente con editar el fichero /etc/inittab y en la sección donde se indica qué programas hay que ejecutar en qué terminales en cada nivel de ejecución añadir una linea para el servidor X, con el tipo de petición que escojamos, en mi sistema Debian esta sección quedaría así:

```
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
7:23:respawn:/usr/bin/X11/X -quiet -query 192.168.0.2 tty7
```

Aunque está fuera del objetivo de esta documentación, y el fichero /etc/inittab tiene su propia página de manual, voy a explicar qué significa cada cosa en estas lineas. El primer número es el número del terminal en el que se va a ejecutar el programa, los numeros que vienen después son los niveles de ejecucion en los que ese programa se ejecutará, lo siquiente indica al sistema qué hacer si el programa muere, respawn le dice al sistema que vuelva a ejecutarlo, por último está el comando a ejecutar. La linea que he añadido es la 7, las opciones que no he comentado hasta ahora son -quiet, que simplemente hace que se muestren menos mensajes por pantalla y tty7 le dice a X que está siendo arrancado por el script init en el dispositivo /dev/tty7. Después de haber añadido esta linea, la proxima vez que el sistema arranque realizará la petición XDMCP y funcionara exactamente como un X Terminal.

También quiero señalar que las dos máquinas no tienen que estar bajo el mismo sistema operativo, un máquina corriendo Linux en un PC puede sin ningún problema ser un X Terminal de un Solaris en una SPARC, que lo he visto funcionando :). Hay incluso servidores X para sistemas Windows. Y con esto pasamos al siguiente apartado.

Xnest y otros

La razón de ser de este apartado no es otra que la de comentar algunas opciones sueltas que tienen que ver con todo lo que hemos comentado hasta ahora, pero no pertenecen a ninguno de los apartados anteriores.

xnest

Xnest es un software curioso. Consiste en unas X anidadas dentro de otras. Por un lado xnest es cliente del servidor X activo, y por otro actua como servidor de X para otras aplicaciones. Si tuviese que simplificarlo mucho diría que es como tener un servidor X a nuestra disposición en una ventanita. Yo lo descubrí desarrollando una live, porque no me valía con chroot, y necesitaba arancar las X de la distro desde el chroot sobre un nuevo display.

A continuación un breve ejemplo de las posibilidades que nos proporciona, después lo dejo en manos de vuestras necesidades e imaginación.

Abrimos un xterm, ejecutamos:

```
jaxvi@portable:~/clase$ Xnest -ac :2
```

En otro xterm, ejecutamos:

```
jaxvi@portable:~$ export DISPLAY=:2.0
jaxvi@portable:~$ xeyes
```

Lo cierto es que hay que darle más vueltas a esto para hacer un buen uso de Xnest, leed el man por ejemplo, y así os ahorraréis ver a vuestras X hacer cosas muuuuy raras :).

Otros

Por último nombraré otras dos maneras que nos pueden ser útiles sobre todo si pretendemos hacer cosas de estas desde windows.

Un primer objetivo podría ser instalar un servidor de X para windows, existen, exceed por ejemplo, y junto con algún cliente ssh como Putty, probar a forwardear las X hacia ese server, que debe ser tan fácil como marcar una opción en putty, teniendo el servidor rulando.

Otra de las que tengo pendientes es hacerlo a través de cygwin. Cygwin es un emulador de linux para correr sobre windows. Se usa mucho y a mi me ha sorprendido las posibilidades que ofrece. Sólo lo he instalado, arrancado e iniciado unas X, lo cual es gracioso en windows. El efecto viene a ser parecido al tema de Xnest, tenemos una ventanita que se supone que son nuestras X. Pues de nuevo, ahora prescindiendo de putty habría que ejecutar ssh -X ..... o X -query ....

Lo que si me parece útil y uso habitualmente es VNC. Virtual Network Computing, es un software de tipo cliente/servidor que se usa para administración de euipos remotamente. A pensar de lo que se piensa (al menos yo pensaba ...) VNC no es un producto de y para windows, en cambio sín llegar a ser protocolo si es cross-plataform, lo que quiere decir en pocas palabras, que podemos conectar por VNC a un windows desde un windows pero también desde un Linux, y al revés, a un Linux desde un Linux o desde un windows también.

Así que (aunque pensaba hacerlo en este documento porque en debian se instala en un apt), ya veremos cómo hacer para en una ventanita de win tener todo nuestro escritorio Linux remoto, mover su puntero del mouse etc etc etc ... Además de curioso puede ser muy útil desde el punto de vista del administrador de sistemas.

Conclusiones

La verdad, a estas alturas tengo la impresión de que he dejado muchas cosas colgando, aunque tampoco puedo permitirme abarcar todas las configuraciones, todos los ejemplos etc ...

Al empezar esta documentación quería hacer algo más completo y ordenado de lo que yo he encontrado por internet. Por un lado teoría sobre lo que es estrictamente xfree86 por ejemplo, o el API de programación etc ... y por otro ejemplos y recetas muy concretas para solucionar las típicas necesidades de usuario, que ocupan 3 líneas. Espero haber encontrado un intermedio, aunque me gustaría perfeccionar este documento en el futuro. El tema es más extenso de lo que parece.

Aunque no he tenido demasiado tiempo siempre es interesante snifar algunas de estas conexiones que realizamos y ver todo esto desde el punto de vista de la seguridad.

Por supuesto, sólo trabajaría con este tipo de cosas en una LAN de confianza, de hecho los propios archivos de configuración desaconsejan estas opciones que vienen deshabilitadas por defecto.

Por otro lado personalmente me ha servido para profundizar y entender mejor como funcionan las X.