LinuxFocus article number 222
http://linuxfocus.org

by Guido Socher (homepage)

*About the author:*

Guido loves Linux not only because it is interesting to understand how operating systems work but also because of the people involved in its design.

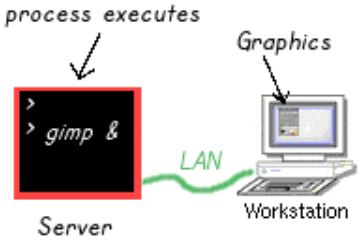
# Running applications remotely with X11

*Abstract*:

Many first time Linux users think that the graphical desktop under Linux is just another "Windows" system where you can start applications and these appear in separate windows. Some people notice that you can have several desktops but that seems to be it. The Linux X Window System (X11) is much more than that! It is a network window system. We will see what new and powerful possibilities this offers.

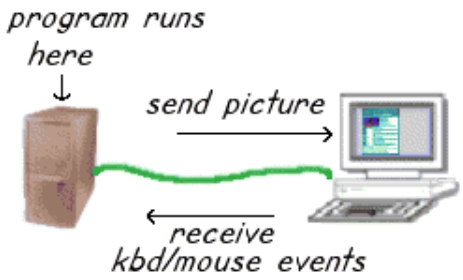
## The display concept

Every graphical X Window application reads at startup the environment variable DISPLAY to find out to which computer screen it should send its graphics. This together with the network capabilities of the X Window System makes it possible to run graphical applications remotely. That is you use the CPU power of one machine while you operate the application from an other one. The entire GUI (graphical user interface) appears on the machine from where you operate it. You don't notice that you use 2 computers.
Network speed is of course an issue here but a normal 10Mbit/s LAN connection is more than enough.




Why would you want to do this?
There are many application of these "network graphics". Companies use it to remotely operate

equipment that might be thousands of kilometers away and you can use the same application to control it as if you would just be at site.
You might have 2 computers a fast 1GHz machine and an old Pentium 133MHz. You can enjoy the speed of your new machine although you are not sitting in front of it. Perhaps you sister is currently sitting in front of the fast machine and logged in. It does not matter you still benefit from it.
How does it work?
All the X Window applications, they may be called gimp, xterm, konquerer, netscape, ... are really network clients that connect to a server, the X-server. The task of the X-server is to talk to the graphics hardware, draw the pictures on your screen, read mouse and keyboard input. The clients (your programs such as gimp, netscape...) send the server instructions on how to paint the frames and buttons. In exchange they receive from the server the mouse and keyboard events. Obviously you need some sort of authentication otherwise everybody could mess up everybody else's screen. There are two programs to control the access:




- xhost: using this program you can allow any user on a given machine to write graphics to your display. Example: You are sitting in front of a machine called philosophus. To allow access for any program on host movietux to your display on philosophus you would type the command:

xhost +movietux
*This must be typed into a shell on philosophus*

- xauth: This is a cookie based authentication and much more sophisticated. Here you can really give individual people access. It is as much more secure then xhost. The authentication uses a cookie stored in the file .Xauthority in the users home directory. If the remote machine has a valid cookie in this file then access will be granted. To copy the cookie from the machine you are sitting in front of (philosophus) to the host from where you want to start the program from (movietux) you can use one of the following methods:

xauth extract - philosophus:0.0 | ssh movietux /usr/X11R6/bin/xauth merge
or
scp ~/.Xauthority movietux:

The procedure that happens at startup of a program (client) is as follows:

1. Client checks the DISPLAY environment variable to find the server otherwise try to connect to the server on this host.
2. Server checks if client is allowed to send "pictures" to him. If the client is authorized then the server will draw the image onto the screen.

The DISPLAY environment variable has the following syntax:

*bash:*export DISPLAY=hostname:displaynumber.screennumber
*tcsh:*setenv DISPLAY hostname:displaynumber.screennumber

I will not talk about the displaynumber and screennumber here. It is normally just 0:0. An example for bash would be:

export DISPLAY=philosophus:0.0

# Starting an application remotely

After all the theory now a practical example. Just try it out. We are again sitting at the computer called philosophus and want to start something remotely from movietux.

1. Tell your X-server that clients from movietux are allowed to draw pictures on your screen:
   xhost +movietux
2. login to the remote host movietux:
   slogin -l username movietux
3. now you are logged on to movietux and every command that you run gets executed on movietux
4. export DISPLAY=philosophus:0.0
5. start the program. E.g:
   netscape &

If movietux is a powerful machine then you will notice that the rendering of html pages in netscape is a lot faster then when you run it locally on your machine. At the same time you don't really notice from usage point of view that this netscape is not started locally because you operate it in the same way with mouse and keyboard.
Cool, isn't it?

# Taking the display with you

Although it is easy to run an application remotely there is still this extra bit of typing needed to set the DISPLAY. It is possible to automate this:

- If you use ssh to login to the remote host then the DISPLAY is automatically set correctly. There are as well other remote login programs that support the DISPLAY but ssh is very common.
- If you use slogin or other commands then you can install the following scripts on the remote host. For tcsh:

  ```
  # take your display with you at remote login:
  # Put it into your ~/.login file
  set whoami=‘who -ml‘
  set remhost=‘expr "$whoami" : ’.*(\(.*\))’‘
  if ( "$remhost" != "" ) then
  setenv DISPLAY "$remhost":0.0
  endif
  ```

  The script works by getting the remote host name from the command "who -ml". This command would return something like

```
>who -ml
movietux!guido pts/3 Oct 26 21:55 (philosophus.tux.org)
```

If you are using bash then you need to the following script:

```
# take your display with you at remote login:
# Put it into your ~/.bash_profile
whoami=‘who -ml‘
remhost=‘expr "$whoami" : ’.*(\(.*\))’‘
if [ -n "$remhost" ]; then
DISPLAY="$remhost":0.0
export DISPLAY
fi
```

# OpenGL

While the networking possibilities of the X Window System are very good, the graphics are a bit slower due to the fact that you send the data over a network protocol. Normally you will not notice much difference.

Graphic intensive and fast applications such as graphic intensive games are usually based on OpenGL (Open Graphics Library) and GLX (OpenGL Extension to the X Window System). These libraries provide a hardware independent programming interface that provides direct access to 3D hardware acceleration in the graphics card. That is: the application sends the description of an object in the form of points, lines and polygons to the graphics card and all the rendering is then done inside the graphics hardware. This provides for very fast graphics.

Currently most Linux graphic card drivers (X servers) do not support hardware-accelerated GLX/OpenGL for remote applications. They do support hardware acceleration for local applications. The effect is that remotely started OpenGL applications are hardly starting at all and are really slow. An exception are the closed source NVidia drivers. They have a direct rendering interface which supports indirect rendering for remote applications.

# Conclusion

Using the computing power of your network is very easy with X11. You can work with remote applications the same way as with local ones. The only difference that you will encounter is that you see the files and home directory on the remote host. However with NFS and NIS installed you can even hide this small difference and use the full CPU power of the fastest machines in your networks without thinking about it.

# Links

X11 system, x.org

xfree86.org the X11 system used with Linux

---



2005-01-14, generated by lfparser_pdf version 2.51