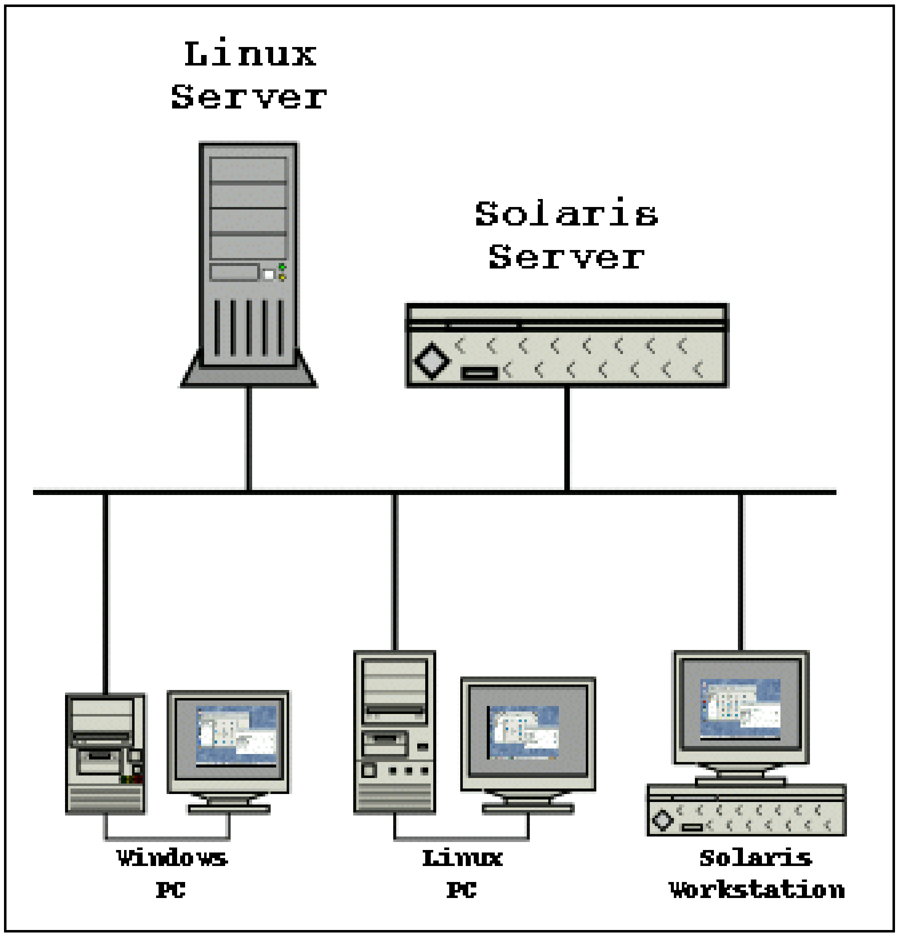
How to enable XDM and VNC for Linux and Solaris

Introduction

XDM is the facility that allows a remote client to access a full desktop on a Unix or Linux server. The protocol it uses is called **XDMCP**. GDM and KDM are the Gnome and KDE enhanced versions of XDM respectively. This functionality can be thought of as equivalent to Windows Terminal Server or Citrix Metaframe on Windows. The downside is that XDMCP does not compress the data. It is comfortably usable over a 10 or 100 Mbit LAN and tolerable over a good broadband connection (e.g. 1 Mbit) but you really do not want to be using it over a dial-up connection. XDM is a standard part of all X11 based Unix or Linux systems. To display an XDM desktop the client machine requires an X server. This is supplied as standard with almost all Unix and Linux systems. X servers are also available for Windows and MacOS.



VNC is another system for viewing remote systems. VNC can be installed so that it connects to the XDM and provides an alternative protocol to XDMCP for accessing it. There other ways of using VNC but this is the most user friendly way so this is what we will concentrate on in this document. VNC has the advantage of being more compressed than X and XDMCP as well as having a much simpler client program. This compression makes VNC just about usable over 56 Kbit dial-up although it is still not as good as RDP or ICA (as used by Windows Terminal Server and Citrix Metaframe respectively). VNC is free software. Multiple implementations exist. Fortunately these are all compatible. VNC clients exist for Unix, Linux, Windows, MacOS, Java and some other systems.

Note: Citrix sells a Metaframe Server for some Unix systems. It is very expensive, but if you regularly need to share Unix desktops over very low bandwidth connections you might find this worthwhile. Try setting up VNC first and if it doesn't meet your needs then you may have to open your wallet to Citrix.

XFS is the X Font Server. This allows the remote client to get fonts from the Unix or Linux machine. Without this the remote desktop will be unable to get the fonts it needs and will look very odd. Without certain fonts some applications may fail completely. XFS is a standard part of any X11 implementation.

None of these facilities is enabled by default on most Linux versions for security reasons. Last time I checked, XDM and XFS were enabled by default on Solaris.

Important: None of these three protocols is secure! You should **not** allow access to them through your firewall. If you want to deploy VNC over the internet you should tunnel it over SSH or some sort of VPN. Doing this is not covered in this document which assumes you are setting this up for use within your LAN or over a private dial-up service.

Enabling XDM Access On Linux

- Edit `/etc/X11/xdm/xdm-config`. Comment out the last line saying:

  ```
  DisplayManager.requestPort: 0
  ```

  by adding an "!" on the front.

- Edit `/etc/X11/xdm/Xaccess`. Uncomment the line:

  ```
  #* #any host can get a login window
  ```

  by removing the first "#".

- The next file to edit depends on which XDM version you are using. For GDM based systems (e.g. RedHat) edit: `/etc/X11/gdm/gdm.conf`. For KDM based systems (e.g. SuSE) edit: `/usr/share/config/kdm/kdmrc`. If unsure, editing both will do no harm.

  In either case, find the section:

  ```
  [Xdmcp]
  Enable=false
  ```

  Change "false" to "true" or "1".

  Underneath this (i.e. somewhere inside the `[Xdmcp]` section), make sure you have a line which says:

  ```
  Port=177
  ```

  If it is commented out then uncomment it. If it says `Port=0` then change it to say `Port=177`. If it is missing entirely then add it.

- You now need to restart the XDM/KDM/GDM process to make it notice the changes. The easiest way to do this is: "`init 3; init 5`". Note that this will crash any X desktop you (or any other users) are running on the machine at the time. If you prefer not to do this, you can kill and restart the processes manually.

Enabling XFS for Linux

- Edit `/etc/X11/fs/config`. Comment out the last line "`no-listen = tcp`" by putting a "#" on the front.
- Restart the font server: /etc/init.d/xfs restart

Installing VNC Server for Linux

Make sure you have tested XDM and XFS first. VNC won't work without them (at least not the way we are going to set it up).

If you don't have the TightVNC RPM to hand get it from: **http://www.tightvnc.com/download.html**. We are going to use TightVNC because it has additional compression features over normal VNC. It is backwards compatible so there won't be any problems with this.

Install the Tight VNC server. Some machines may already have the standard VNC server. You can install Tight VNC over the top as an upgrade:

e.g. `rpm -Uvh tightvnc-server-1.2.8-1.i386.rpm`

Installing VNC Server for Solaris

I couldn't get any sense out of TightVNC 1.2.8 when I tried to compile it on Solaris. Instead, I have found RealVNC to be more tractable. RealVNC is compatible with TightVNC and VNC but lacks a few of TightVNC's cleverer features. Chances are nobody will notice the difference.

If you don't have it, get the Solaris binary version of RealVNC from: **http://www.realvnc.com/download.html**

Note: The binary says it is made for Solaris 2.5 but it works OK on Solaris 8. Unfortunately it is only 32 bit.

Unpack the binary in /opt. You don't need to bother with the install script if you don't want to. I just did the following:

```
cd /opt
gunzip vnc-3.3.6-sparc_solaris_2.5.tar.gz
tar -xvf vnc-3.3.6-sparc_solaris_2.5.tar
ln -s vnc-3.3.6-sparc_solaris_2.5 vnc
```

Enabling VNC Server for Linux and Solaris

We now need to add VNC services to the machine. Each one will connect to an XDM display running a fixed resolution and colour depth. To give the users plenty of choice we can make several of these. The users can then connect to whichever port corresponds to the resolution they want.

Edit `/etc/services` and add the services you want. e.g:

```
vnc-640x480x8   5950/tcp                       # VNC :50 -
Recommended for low bandwidth users.
vnc-800x600x8   5951/tcp                       # VNC :51
vnc-800x600x16  5952/tcp                       # VNC :52
vnc-800x600x24  5953/tcp                       # VNC :53
vnc-1024x768x8  5954/tcp                       # VNC :54
vnc-1024x768x16 5955/tcp                       # VNC :55 -
Recommended for most users.
vnc-1024x768x24 5956/tcp                       # VNC :56
vnc-1152x864x8  5957/tcp                       # VNC :57
vnc-1152x864x16 5958/tcp                       # VNC :58
vnc-1152x864x24 5959/tcp                       # VNC :59
vnc-1280x1024x8 5960/tcp                       # VNC :60
vnc-1280x1024x16 5961/tcp                      # VNC :61 -
Recommended for power users.
vnc-1280x1024x24 5962/tcp                      # VNC :62
```

Systems Using Inetd

Solaris and some Linux versions use the traditional inetd service. Edit `/etc/inetd.conf` and add entries which correspond to the services you put in /etc/services. e.g:

```
#
# VNC Server
#
vnc-640x480x8    stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 640x480 -depth 8
vnc-800x600x8    stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 800x600 -depth 8
vnc-800x600x16   stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 800x600 -depth 16
vnc-800x600x24   stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 800x600 -depth 24
vnc-1024x768x8   stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
```

```
localhost -fp tcp/localhost:7100 -once -geometry 1024x768 -depth 8
vnc-1024x768x16  stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1024x768 -depth 16
vnc-1024x768x24  stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1024x768 -depth 24
vnc-1152x864x8   stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1152x864 -depth 8
vnc-1152x864x16  stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1152x864 -depth 16
vnc-1152x864x24  stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1152x864 -depth 24
vnc-1280x1024x8  stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1280x1024 -depth 8
vnc-1280x1024x16 stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1280x1024 -depth 16
vnc-1280x1024x24 stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query
localhost -fp tcp/localhost:7100 -once -geometry 1280x1024 -depth 24
```

Find the process for inetd and kill -HUP it:

```
ps -ef | grep inetd
kill -HUP pid
```

Note: You should run the VNC services as nobody for the standard security reasons. On Solaris I was unable to make it run as nobody and had to change it to root. If you have the same problem then you will need to decide whether to take this risk.

Systems Using Xinetd

RedHat and some other Linux systems use the new Xinetd service instead of Inetd. This is functionally similar but has a more verbose file format.

Make a new file in `/etc/xinetd.d` called `vnc` and put entries in it which correspond to the services you put in /etc/services. e.g:

```
# default: on
# description: VNC.

service vnc-640x480x8
{
        protocol = tcp
        socket_type     = stream
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -inetd -query localhost -fp tcp/localhost:7100 -once -
geometry 640x480 -depth 8
}

service vnc-800x600x8
{
        protocol = tcp
        socket_type     = stream
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -inetd -query localhost -fp tcp/localhost:7100 -once -
geometry 800x600 -depth 8
}

service vnc-800x600x16
{
        protocol = tcp
        socket_type     = stream
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -inetd -query localhost -fp tcp/localhost:7100 -once -
geometry 800x600 -depth 16
}

service vnc-800x600x24
{
        protocol = tcp
        socket_type     = stream
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -inetd -query localhost -fp tcp/localhost:7100 -once -
geometry 800x600 -depth 24
}

And so on.......
```

Restart Xinetd:

```
/etc/init.d/xinetd restart
```

Digression: A Clever Trick

You may be wondering about the fact that we had to tell VNC to connect itself to the XDM on the same machine by telling it to query `localhost`. You may be wondering whether you could put other machine names in there. Yes you can, provided that the other machine is also serving XDMCP sessions. This is pretty cool. It means that you can have a single

VNC server serving VNC sessions for many machines. There are a number of reasons you might want to do this:

- To save the effort administering VNC servers on multiple machines.
- To move the CPU load of VNC serving off your main server(s) and on to a dedicated machine.
- To support TightVNC access to flavours of Unix which can not easily run TightVNC.
- To show off how clever you are. ;-)

Lets look at a quick example. Suppose that, in addition to running VNC displays 50 to 62 aimed at itself, we want to make VNC connections to a second server which is imaginatively titled serverb. Let us assume that serverb is already serving XDMCP. We don't need to do anything with serverb. On the original server we add one or more additional services in `/etc/services` each on a new port number. e.g:

```
vnc-serverb 5970/tcp                      # VNC :70
```

Now we add a corresponding VNC service to inetd or xinetd. e.g:

```
vnc-serverb stream tcp nowait nobody /opt/vnc/Xvnc Xvnc -inetd -query serverb -fp
tcp/serverb:7100 -once -geometry 1024x768 -depth 16
```

Restart inetd or xinetd and that is it. To use it make a VNC connection to the original server, not serverb. E.g. to access serverb you might connect to `servera:70`. Weird maybe, but it works.

Security Issues on Linux

You don't want remote users shutting down the machine. Remote VNC users appear to be local as far as X is concerned and hence will have the option to shut the machine down from KDE and Gnome. To get round this, you need to stop users other than root shutting down the machine.

Log into KDE as root. Bring up the Control Centre and go to the Login Manager section. Set it so that only root can shutdown the machine (either locally or remotely).

Finally, to stop users shutting down from Gnome or a shell do: `chmod 700 /sbin/shutdown`

Testing

Note: Replace "*hostname*" with the name of your server throughout this section.

Testing XDM

To check whether XDM is working from a Linux workstation run:

```
X -query hostname :1
```

This will start a second desktop (assuming that you already have an X desktop running). Swap between them with CTRL-ALT-F7 and CRTL-ALT-F8. You can also abort the X displays with CTRL-ALT-Backspace.

To check whether XDM is working from a Solaris workstation drop out of graphics mode (`init 2`) and do:

```
X -query hostname :0
```

To check whether XDM is working from a Windows PC you will need an X server installed. If you have Cygwin/XFree86 then do:

```
X -query hostname :0
```

In any of the above cases, you should see an X desktop appear showing the login screen of the server in the same way as if you were using it locally. The fonts may appear odd. Getting the fonts right is the next stage.

Testing XFS

Once you have XDM working, you should check the font server. Quit any XDM session left over from before and invoke XDM again but this time tell it to use the font server:

```
X -query hostname -fp tcp/hostname:7100 :1
```

or:

```
X -query hostname -fp tcp/hostname:7100 :0
```

This should work as before. The fonts should now look more or less correct but may not be perfect. You won't get anti-aliasing and some fonts may appear imperfectly scaled (i.e. blocky). Try logging in. Run a few applications. They should all be usable.

Testing VNC

To test VNC from a Linux or Solaris workstation, make sure that a VNC client is installed and do:

```
vncviewer hostname:display
```

where *display* is one of the VNC display numbers you set up on the server (i.e. 50 to 62 in our example).

To test from Windows, make sure that the TightVNC client is installed and then try connecting to one of the VNC displays which you set up on the server using the TightVNC Viewer (Fast Compression) option.

Using VNC

If all went well you should now have a usable service. You will need to install the appropriate VNC clients on all the users' PCs/Workstations and then tell the users which servers and displays are provided.

Note: VNC and X display names are not related despite their similar format of "hostname: number". You can have multiple users on each VNC display service. If several users all connect to VNC display hostname:55 then it will start a new and separate X display for each one (i.e. localhost:1, localhost:2, etc). This means that you can have as many concurrent users as you like provided the server has enough CPU and RAM.

Dial-up users will find the service sluggish. This can be optimised with a few tricks, although it will never be fast:

- Use maximum compression in the VNC viewer.
- Use the TightVNC version of the VNC Viewer whenever possible. This has extra compression features.
- Use a low resolution VNC display and avoid 24 bit colour.
- Turn off all animations, wallpaper and other "eye candy" on the remote desktop.
- Consider using a more basic Window Manager than your normal one. TWM, FVWM or OpenWindows are much less graphically intensive than KDE, GNOME or CDE.

Latest update: 05/05/2003.
Many thanks to Damian McGuckin for suggesting improvements.