Xorg

Related articles

- Autostarting
- Cursor themes
- Desktop environment
- Display manager
- Font configuration
- Window manager
- XDMCP
- xinit
- xrandr

X.Org Server — commonly referred to as simply **X** — is the X.Org Foundation implementation of the X Window System (**X11**) display server, and it is the most popular display server among Linux users. Its ubiquity has led to making it an ever-present requisite for GUI applications, resulting in massive adoption from most distributions.

For the alternative and successor, see Wayland.

1 Installation

Xorg can be installed with the `xorg-server (https://archlinux.org/packages/?name=xorg-server)` package.

Additionally, some packages from the `xorg-apps (https://archlinux.org/groups/x86_64/xorg-apps/)` group are necessary for certain configuration tasks. They are pointed out in the relevant sections.

Finally, an `xorg (https://archlinux.org/groups/x86_64/xorg/)` group is also available, which includes Xorg server packages, packages from the `xorg-apps (https://archlinux.org/groups/x86_64/xorg-apps/)` group and fonts.

1.1 Driver installation

The Linux kernel includes open-source video drivers and support for hardware accelerated framebuffers. However, userland support is required for OpenGL, Vulkan and 2D acceleration in X11.

First, identify the graphics card (the *Subsystem* output shows the specific model):

```
$ lspci -v -nn -d ::03xx
```

Tip

`::03` here means "Display controller (https://admin.pci-ids.ucw.cz/read/PD/03) PCI device class", and `xx` stands for "any subclass of the class".

Then, install an appropriate driver. You can search the package database for a complete list of open-source Device Dependent X (DDX) (https://dri.freedesktop.org/wiki/DDX/) drivers:

```
$ pacman -Ss xf86-video
```

However, hardware-specific DDX is considered legacy nowadays. There is a generic modesetting(4) (https://man.archlinux.org/man/modesetting.4) DDX driver in xorg-server (https://archlinux.org/packages/?name=xorg-server), which uses kernel mode setting and works well on modern hardware. The modesetting DDX driver uses Glamor (https://www.freedesktop.org/wiki/Software/Glamor/)[1] (https://gitlab.freedesktop.org/xorg/xserver/-/tree/server-21.1-branch/glamor) for 2D acceleration, which requires OpenGL.

If you want to install another DDX driver, note that Xorg searches for installed DDX drivers automatically:

- If it cannot find the specific driver installed for the hardware (listed below), it first searches for *fbdev* (xf86-video-fbdev (https://archlinux.org/packages/?name=xf86-video-fbdev)), which does not include any 2D or 3D acceleration.
- If that is not found, it searches for *vesa* (xf86-video-vesa (https://archlinux.org/packages/?name=xf86-video-vesa)), the generic driver, which handles a large number of chipsets but does not include any 2D or 3D acceleration.
- If *vesa* is not found, Xorg will fall back to modesetting(4) (https://man.archlinux.org/man/modesetting.4) DDX driver.

In order for video acceleration to work, and often to expose all the modes that the GPU can set, a proper video driver is required:

Brand	Type	Documentation	DRM driver	OpenGL	OpenGL (multilib)	Vulkan	Vulkan (multilib)	DDX driver
AMD (ex-ATI)	Open source	AMDGPU	included in Linux	mesa (https://archlinux.org/packages/?name=mesa)[3]	lib32-mesa (https://archlinux.org/packages/?name=lib32-mesa)[3]	vulkan-radeon (https://archlinux.org/packages/?name=vulkan-radeon) / amdvlk (https://archlinux.org/packages/?name=amdvlk)[4]	lib32-vulkan-radeon (https://archlinux.org/packages/?name=lib32-vulkan-radeon) / lib32-amdvlk (https://archlinux.org/packages/?name=lib32-amdvlk)[4]	xf86-video-amdgpu (https://archlinux.org/packages/?name=xf86-video-amdgpu)
		ATI				None		xf86-video-ati (https://archlinux.org/packages/?name=xf86-video-ati)
	Proprietary	AMDGPU PRO		amdgpu-pro-oglp (https://aur.archlinux.org/packages/amdgpu-pro-oglp/)AUR	lib32-amdgpu-pro-oglp (https://aur.archlinux.org/packages/lib32-amdgpu-pro-oglp/)AUR	vulkan-amdgpu-pro (https://aur.archlinux.org/packages/vulkan-amdgpu-pro/)AUR	lib32-vulkan-amdgpu-pro (https://aur.archlinux.org/packages/lib32-vulkan-amdgpu-pro/)AUR	xf86-video-amdgpu (https://archlinux.org/packages/?name=xf86-video-amdgpu)
Intel	Open source	Intel graphics		mesa (https://archlinux.org/package	lib32-mesa (https://archlinux.org/package	vulkan-intel (https://archlinux.org/pac	lib32-vulkan-intel (https://archlinux.	xf86-video-intel (https://archlinux.

				s/?name=mesa)[3]	s/?name=lib32-mesa)[3]	kages/?name=vulkan-intel)	org/packages/?name=lib32-vulkan-intel)	org/packages/?name=xf86-video-intel)[2]
NVIDIA	Open source	Nouveau[1]		mesa (https://archlinux.org/packages/?name=mesa)[3]	lib32-mesa (https://archlinux.org/packages/?name=lib32-mesa)[3]	vulkan-nouveau (https://archlinux.org/packages/?name=vulkan-nouveau)	lib32-vulkan-nouveau (https://archlinux.org/packages/?name=lib32-vulkan-nouveau)	xf86-video-nouveau (https://archlinux.org/packages/?name=xf86-video-nouveau)
	Proprietary	NVIDIA[1]	nvidia (https://archlinux.org/packages/?name=nvidia) or nvidia-open (https://archlinux.org/packages/?name=nvidia-open)[5]	nvidia-utils (https://archlinux.org/packages/?name=nvidia-utils)	lib32-nvidia-utils (https://archlinux.org/packages/?name=lib32-nvidia-utils)	nvidia-utils (https://archlinux.org/packages/?name=nvidia-utils)	lib32-nvidia-utils (https://archlinux.org/packages/?name=lib32-nvidia-utils)	nvidia-utils (https://archlinux.org/packages/?name=nvidia-utils)
			nvidia-535xx-dkms (https://aur.archlinux.org/packages/nvidia-535xx-dkms/)AUR	nvidia-535xx-utils (https://aur.archlinux.org/packages/nvidia-535xx-utils/)AUR	lib32-nvidia-535xx-utils (https://aur.archlinux.org/packages/lib32-nvidia-535xx-utils/)AUR	nvidia-535xx-utils (https://aur.archlinux.org/packages/nvidia-535xx-utils/)AUR	lib32-nvidia-535xx-utils (https://aur.archlinux.org/packages/lib32-nvidia-535xx-utils/)AUR	nvidia-535xx-utils (https://aur.archlinux.org/packages/nvidia-535xx-utils/)AUR
			nvidia-470xx-dkms (https://aur.arc	nvidia-470xx-utils (https://au	lib32-nvidia-470xx-utils (http	nvidia-470xx-utils (https://au	lib32-nvidia-470xx-utils (http	nvidia-470xx-utils (https://au

			hlinux.org/packages/nvidia-470xx-dkms/)$^{\text{AUR}}$	r.archlinux.org/packages/nvidia-470xx-utils/)$^{\text{AUR}}$	s://aur.archlinux.org/packages/lib32-nvidia-470xx-utils/)$^{\text{AUR}}$	r.archlinux.org/packages/nvidia-470xx-utils/)$^{\text{AUR}}$	s://aur.archlinux.org/packages/lib32-nvidia-470xx-utils/)$^{\text{AUR}}$	r.archlinux.org/packages/nvidia-470xx-utils/)$^{\text{AUR}}$
			nvidia-390xx-dkms (https://aur.archlinux.org/packages/nvidia-390xx-dkms/)$^{\text{AUR}}$	nvidia-390xx-utils (https://aur.archlinux.org/packages/nvidia-390xx-utils/)$^{\text{AUR}}$	lib32-nvidia-390xx-utils (https://aur.archlinux.org/packages/lib32-nvidia-390xx-utils/)$^{\text{AUR}}$	nvidia-390xx-utils (https://aur.archlinux.org/packages/nvidia-390xx-utils/)$^{\text{AUR}}$	lib32-nvidia-390xx-utils (https://aur.archlinux.org/packages/lib32-nvidia-390xx-utils/)$^{\text{AUR}}$	nvidia-390xx-utils (https://aur.archlinux.org/packages/nvidia-390xx-utils/)$^{\text{AUR}}$

1. For NVIDIA Optimus enabled laptop which uses an integrated video card combined with a dedicated GPU, see **NVIDIA Optimus**.
2. For Intel graphics on 4th generation and above, the *modesetting* DDX driver is recommended. See **Intel graphics#Installation** for details.
3. For older hardware, classic OpenGL (non-Gallium3D) drivers in mesa-amber (https://archlinux.org/packages/?name=mesa-amber)/lib32-mesa-amber (https://archlinux.org/packages/?name=lib32-mesa-amber) might be useful (Mesa 22.0 and higher have dropped support for non-Gallium3D classic drivers), see **OpenGL#Installation**.
4. vulkan-radeon (https://archlinux.org/packages/?name=vulkan-radeon) / lib32-vulkan-radeon (https://archlinux.org/packages/?name=lib32-vulkan-radeon) is recommended over amdvlk (https://archlinux.org/packages/?name=amdvlk) / lib32-amdvlk (https://archlinux.org/packages/?name=lib32-amdvlk) (see **AMDGPU#Installation**).
5. For the difference between nvidia (https://archlinux.org/packages/?name=nvidia) and nvidia-open (https://archlinux.org/packages/?name=nvidia-open), see **NVIDIA#Installation**.

Other DDX drivers can be found in the xorg-drivers (https://archlinux.org/groups/x86_64/xorg-drivers/) group.

Xorg should run smoothly without closed source drivers, which are typically needed only for advanced features such as fast 3D-accelerated rendering for games. The exceptions to this rule are recent GPUs (especially NVIDIA GPUs) not supported by open source drivers.

1.1.1 AMD

For a translation of model names (e.g. *Radeon RX 6800*) to GPU architectures (e.g. *RDNA 2*), see **Wikipedia:List of AMD graphics processing units#Features overview**.

GPU architecture	Open-source driver	Proprietary driver
RDNA and later	AMDGPU	AMDGPU PRO
GCN 3 and later		
GCN 1&2	AMDGPU[1] / ATI	*not available*
TeraScale and older	ATI	*not available*

1. Experimental.

2 Running

The Xorg(1) (https://man.archlinux.org/man/Xorg.1) command is usually not run directly. Instead, the X server is started with either a **display manager** or **xinit**.

Tip

You will typically seek to install a **window manager** or a **desktop environment** to supplement X.

3 Configuration

Note

Arch supplies default configuration files in `/usr/share/X11/xorg.conf.d/` , and no extra configuration is necessary for most setups.

Xorg uses a configuration file called `xorg.conf` and files ending in the suffix `.conf` for its initial setup: the complete list of the folders where these files are searched can be found in xorg.conf(5) (https://man.archlinux.org/man/xorg.conf.5), together with a detailed explanation of all the available options.

3.1 Using .conf files

The `/etc/X11/xorg.conf.d/` directory stores host-specific configuration. You are free to add configuration files there, but they must have a `.conf` suffix: the files are read in ASCII order, and by convention their names start with *`XX-`* (two digits and a hyphen, so that for example 10 is read before 20). These files are parsed by the X server upon startup and are treated like part of the traditional `xorg.conf` configuration file. Note that on conflicting configuration, the file read *last* will be processed. For this reason, the most generic configuration files should be ordered first by name. The configuration entries in the `xorg.conf` file are processed at the end.

For option examples to set, see **Fedora:Input device configuration#xorg.conf.d**.

3.2 Using xorg.conf

Xorg can also be configured via `/etc/X11/xorg.conf` or `/etc/xorg.conf`. You can also generate a skeleton for `xorg.conf` with:

```
# Xorg :0 -configure
```

This should create a `xorg.conf.new` file in `/root/` that you can copy over to `/etc/X11/xorg.conf`.

Tip

If you are already running an X server, use a different display, for example `Xorg :2 -configure`.

Alternatively, your proprietary video card drivers may come with a tool to automatically configure Xorg: see the article of your video driver, **NVIDIA** or **AMDGPU PRO**, for more details.

Note

Configuration file keywords are case insensitive, and "_" characters are ignored. Most strings (including Option names) are also case insensitive, and insensitive to white space and "_" characters.

4 Input devices

For input devices the X server defaults to the libinput driver (`xf86-input-libinput (https://archlinux.org/packages/?name=xf86-input-libinput)`), but `xf86-input-evdev (https://archlinux.org/packages/?name=xf86-input-evdev)` and related drivers are available as alternative.[2] (https://archlinux.org/news/xorg-server-1191-is-now-in-extra/)

Udev, which is provided as a systemd dependency, will detect hardware and both drivers will act as hotplugging input driver for almost all devices, as defined in the default configuration files `10-quirks.conf` and `40-libinput.conf` in the `/usr/share/X11/xorg.conf.d/` directory.

After starting X server, the log file will show which driver hotplugged for the individual devices (note the most recent log file name may vary):

```
$ grep -e "Using input driver " Xorg.0.log
```

If both do not support a particular device, install the needed driver from the `xorg-drivers (https://archlinux.org/groups/x86_64/xorg-drivers/)` group. The same applies, if you want to use another driver.

To influence hotplugging, see **#Configuration**.

For specific instructions, see also the **libinput** article, the following pages below, or **Fedora:Input device configuration** for more examples.

4.1 Input identification

See **Keyboard input#Identifying keycodes in Xorg**.

4.2 Mouse acceleration

See **Mouse acceleration**.

4.3 Extra mouse buttons

See **Mouse buttons**.

4.4 Touchpad

See **libinput** or **Synaptics**.

4.5 Touchscreen

See **Touchscreen**.

4.6 Keyboard settings

See **Keyboard configuration in Xorg**.

5 Monitor settings

5.1 Manual configuration

Note

- Newer versions of Xorg are auto-configuring, so manual configuration should not be needed.
- If Xorg is unable to detect any monitor or to avoid auto-configuring, a configuration file can be used. A common case where this is necessary is a headless system, which boots without a monitor and starts Xorg automatically, either from a **virtual console** at **login**, or from a **display manager**.

For a headless configuration, the `xf86-video-dummy (https://archlinux.org/packages/?name=xf86-video-dummy)` driver is necessary; **install** it and create a configuration file, such as the following:

```
/etc/X11/xorg.conf.d/10-headless.conf
```

```
Section "Monitor"
        Identifier "dummy_monitor"
        HorizSync 28.0-80.0
        VertRefresh 48.0-75.0
        Modeline "1920x1080" 172.80 1920 2040 2248 2576 1080 1081 1084 1118
```

```
EndSection

Section "Device"
        Identifier "dummy_card"
        VideoRam 256000
        Driver "dummy"
EndSection

Section "Screen"
        Identifier "dummy_screen"
        Device "dummy_card"
        Monitor "dummy_monitor"
        SubSection "Display"
        EndSubSection
EndSection
```

5.2 Multiple monitors

See main article **Multihead** for general information.

5.2.1 More than one graphics card

You must define the correct driver to use and put the bus ID of your graphic cards (in decimal notation).

```
Section "Device"
    Identifier             "Screen0"
    Driver                 "intel"
    BusID                  "PCI:0:2:0"
EndSection

Section "Device"
    Identifier             "Screen1"
    Driver                 "nouveau"
    BusID                  "PCI:1:0:0"
EndSection
```

To get your bus IDs (in hexadecimal):

```
$ lspci -d ::03xx
```

```
00:02.0 VGA compatible controller: Intel Corporation HD Graphics 630 (rev 04)
01:00.0 3D controller: NVIDIA Corporation GP107M [GeForce GTX 1050 Mobile] (rev a1)
```

The bus IDs here are `0:2:0` and `1:0:0`.

5.3 Display size and DPI

By default, Xorg always sets DPI to 96 since **2009-01-30 (https://gitlab.freedesktop.org/xorg/xserver/-/commit/fff00df94d7ebd18a8e24537ec96073717375a3f)**. A change was made with version 21.1 to provide proper DPI auto-detection, but **reverted (https://gitlab.freedesktop.org/xorg/xserver/-/commit/35af1299e73483eaf93d913a960e1d1738bc7de6)**.

The DPI of the X server can be set with the `-dpi` command line option.

Having the correct DPI is helpful where fine detail is required (like font rendering). Previously, manufacturers tried to create a standard for 96 DPI (a 10.3" diagonal monitor would be 800x600, a 13.2" monitor 1024x768). These days, screen DPIs vary and may not be equal horizontally and vertically. For example, a 19" widescreen

LCD at 1440x900 may have a DPI of 89x87.

To see if your display size and DPI are correct:

```
$ xdpyinfo | grep -B2 resolution
```

Check that the dimensions match your display size.

If you have specifications on the physical size of the screen, they can be entered in the Xorg configuration file so that the proper DPI is calculated (adjust identifier to your xrandr output):

```
Section "Monitor"
    Identifier             "DVI-D-0"
    DisplaySize             286 179    # In millimeters
EndSection
```

If you only want to enter the specification of your monitor **without** creating a full xorg.conf, create a new configuration file. For example (`/etc/X11/xorg.conf.d/90-monitor.conf`):

```
Section "Monitor"
    Identifier             "<default monitor>"
    DisplaySize            286 179    # In millimeters
EndSection
```

Note

If you are using the proprietary NVIDIA driver, you may have to put `Option "UseEdidDpi" "FALSE"` under `Device` or `Screen` section to make it take effect.

If you do not have specifications for physical screen width and height (most specifications these days only list by diagonal size), you can use the monitor's native resolution (or aspect ratio) and diagonal length to calculate the horizontal and vertical physical dimensions. Using the Pythagorean theorem on a 13.3" diagonal length screen with a 1280x800 native resolution (or 16:10 aspect ratio):

```
$ echo 'scale=5;sqrt(1280^2+800^2)' | bc  # 1509.43698
```

This will give the pixel diagonal length, and with this value you can discover the physical horizontal and vertical lengths (and convert them to millimeters):

```
$ echo 'scale=5;(13.3/1509)*1280*25.4' | bc  # 286.43072
$ echo 'scale=5;(13.3/1509)*800*25.4'  | bc  # 179.01920
```

Note

This calculation works for monitors with square pixels; however, there is the rare monitor that may compress aspect ratio (e.g 16:10 aspect resolution to a 16:9 monitor). If this is the case, you should measure your screen size manually.

5.3.1 Setting DPI manually

Note

While you can set any DPI you like and applications using Qt and GTK will scale accordingly, it is recommended to set it to **96** (100%, no scaling), **120** (25% higher), **144** (50% higher), **168** (75% higher), **192** (100% higher) etc., to reduce scaling artifacts to GUIs that use bitmaps. Reducing it below 96 DPI may not reduce the size of the GUIs graphical elements, as typically the lowest DPI the icons are made for is 96.

For RandR compliant drivers (for example the open source ATI driver), you can set it by:

```
$ xrandr --dpi 144
```

Note

Applications that comply with the setting will not change immediately. You have to start them anew.

To make it permanent, see **Autostarting#On Xorg startup**.

5.3.1.1 Proprietary NVIDIA driver

You can manually set the DPI by adding the option under the `Device` or `Screen` section:

```
Option              "DPI" "96 x 96"
```

5.3.1.2 Manual DPI Setting Caveat

GTK very often overrides the server's DPI via the optional **X resource** `Xft.dpi`. To find out whether this is happening to you, check with:

```
$ xrdb -query | grep dpi
```

With GTK library versions since 3.16, when this variable is not otherwise explicitly set, GTK sets it to 96. To have GTK apps obey the server DPI you may need to explicitly set `Xft.dpi` to the same value as the server. The `Xft.dpi` resource is the method by which some desktop environments optionally force DPI to a particular value in personal settings. Among these are **KDE** and **TDE**.

5.4 Display Power Management

DPMS is a technology that allows power saving behaviour of monitors when the computer is not in use. This will allow you to have your monitors automatically go into standby after a predefined period of time.

6 Composite

The Composite extension for X causes an entire sub-tree of the window hierarchy to be rendered to an off-screen buffer. Applications can then take the contents of that buffer and do whatever they like. The off-screen buffer can be automatically merged into the parent window, or merged by external programs called compositing managers. For more information, see **Wikipedia:Compositing window manager**.

Some window managers (e.g. **Compiz**, **Enlightenment**, **KWin**, `marco (https://archlinux.org/packages/?name=marco)`, `metacity (https://archlinux.org/packages/?name=metacity)`, `muffin (https://archlinux.org/packages/?name=muffin)`, `mutter (https://archlinux.org/packages/?name=mutter)`, **Xfwm**) do compositing on their own. For other window managers, a standalone composite manager can be used.

6.1 List of composite managers

- **Picom** — Lightweight compositor with shadowing, advanced blurring and fading. Forked from Compton.

  https://github.com/yshui/picom || `picom (https://archlinux.org/packages/?name=picom)`

- **Xcompmgr** — Composite window-effects manager.

  https://gitlab.freedesktop.org/xorg/app/xcompmgr/ || `xcompmgr (https://archlinux.org/packages/?name=xcompmgr)`

- **Gamescope** — The micro-compositor from Valve, with gaming-oriented features such as FSR upscaling. Forked from steamos-compositor.

  https://github.com/ValveSoftware/gamescope || `gamescope (https://archlinux.org/packages/?name=gamescope)`

- **steamos-compositor-plus** — Valve's compositor, with some added tweaks and fixes.

  https://github.com/chimeraos/steamos-compositor-plus || `steamos-compositor-plus (https://aur.archlinux.org/packages/steamos-compositor-plus/)`AUR

7 Tips and tricks

7.1 Automation

This section lists utilities for automating keyboard / mouse input and window operations (like moving, resizing or raising).

Tool	Package	Manual	Keysym input	Window operations	Note
xautomation	xautomation (https://archlinux.org/packages/?name=xautomation)	xte(1) (https://man.archlinux.org/man/xte.1)	Yes	No	Also contains screen scraping tools. Cannot simulate `F13` and more.
xdo	xdo (https://archlinux.org/packages/?name=xdo)	xdo(1) (https://man.archlinux.org/man/xdo.1)	No	Yes	Small X utility to perform elementary actions on windows.
xdotool	xdotool (https://archlinux.org/packages/?name=xdotool)	xdotool(1) (https://man.archlinux.org/man/xdotool.1)	Yes	Yes	Very buggy (https://github.com/jordansissel/xdotool/issues) and not in active development, e.g: has broken CLI parsing.[3] (https://github.com/jordansissel/xdotool/issues/14#issuecomment-327968132) [4] (https://github.com/jordansissel/xdotool/issues/71)
xvkbd	xvkbd (https://aur.archlinux.org/packages/xvkbd/) AUR	xvkbd(1) (http://t-sato.in.coocan.jp/xvkbd/#option)	Yes	No	Virtual keyboard for Xorg, also has the `-text` option for sending characters.
AutoKey	autokey-qt (https://aur.archlinux.org/packages/autokey-qt/) AUR autokey-gtk (https://aur.archlinux.org/packages/autokey-gtk/) AUR	documentation (https://github.com/autokey/autokey#documentation)	Yes	Yes	Higher-level, powerful macro and scripting utility, with both Qt and Gtk front-ends.

See also Clipboard#Tools and an overview of X automation tools (https://venam.nixers.net/blog/unix/2019/01/07/win-automation.html).

7.2 Nested X session

To run a nested session of another desktop environment:

```
$ /usr/bin/Xnest :1 -geometry 1024x768+0+0 -ac -name Windowmaker & wmaker -display :1
```

This will launch a Window Maker session in a 1024 by 768 window within your current X session.

This needs the package xorg-server-xnest (https://archlinux.org/packages/?name=xorg-server-xnest) to be installed.

A more modern way of doing a nested X session is with Xephyr.

7.3 Starting an application without a window manager

See xinit#Starting applications without a window manager.

7.4 Starting GUI programs remotely

See main article: OpenSSH#X11 forwarding.

7.5 On-demand disabling and enabling of input sources

With the help of *xinput* you can temporarily disable or enable input sources. This might be useful, for example, on systems that have more than one mouse, such as the ThinkPads and you would rather use just one to avoid unwanted mouse clicks.

Install the `xorg-xinput` (https://archlinux.org/packages/?name=xorg-xinput) package.

Find the name or ID of the device you want to disable:

```
$ xinput
```

For example in a Lenovo ThinkPad T500, the output looks like this:

```
$ xinput
```

```
⎡ Virtual core pointer                          id=2    [master pointer  (3)]
⎜   ↳ Virtual core XTEST pointer                id=4    [slave  pointer  (2)]
⎜   ↳ TPPS/2 IBM TrackPoint                     id=11   [slave  pointer  (2)]
⎜   ↳ SynPS/2 Synaptics TouchPad                id=10   [slave  pointer  (2)]
⎣ Virtual core keyboard                         id=3    [master keyboard (2)]
    ↳ Virtual core XTEST keyboard               id=5    [slave  keyboard (3)]
    ↳ Power Button                              id=6    [slave  keyboard (3)]
    ↳ Video Bus                                 id=7    [slave  keyboard (3)]
    ↳ Sleep Button                              id=8    [slave  keyboard (3)]
    ↳ AT Translated Set 2 keyboard              id=9    [slave  keyboard (3)]
    ↳ ThinkPad Extra Buttons                    id=12   [slave  keyboard (3)]
```

Disable the device with `xinput --disable device`, where *device* is the device ID or name of the device you want to disable. In this example we will disable the Synaptics Touchpad, with the ID 10:

```
$ xinput --disable 10
```

To re-enable the device, just issue the opposite command:

```
$ xinput --enable 10
```

Alternatively using the device name, the command to disable the touchpad would be:

```
$ xinput --disable "SynPS/2 Synaptics TouchPad"
```

7.6 Persistently disable input source

You can disable a particular input source using a configuration snippet:

```
/etc/X11/xorg.conf.d/30-disable-device.conf
```

```
Section "InputClass"
	Identifier   "disable-device"
	Driver       "driver_name"
	MatchProduct "device_name"
	Option       "Ignore" "True"
EndSection
```

`device` is an arbitrary name, and `driver_name` is the name of the input driver, e.g. `libinput`. `device_name` is what is actually used to match the proper device. For alternate methods of targeting the correct device, such as **libinput**'s `MatchIsTouchscreen`, consult your input driver's documentation. Though this example uses libinput, this is a driver-agnostic method which simply prevents the device from being propagated to the driver.

7.7 Killing application with hotkey

Run script on hotkey:

```
#!/bin/sh
windowFocus=$(xdotool getwindowfocus)
pid=$(xprop -id "$windowFocus" | grep PID)
kill -9 "$pid"
```

Dependencies: `xorg-xprop (https://archlinux.org/packages/?name=xorg-xprop)`, `xdotool (https://archlinux.org/packages/?name=xdotool)`

See also **#Killing an application visually**.

7.8 Block TTY access

To block tty access when in an X add the following to **xorg.conf**:

```
Section "ServerFlags"
    Option "DontVTSwitch" "True"
EndSection
```

This can be used to help restrict command line access on a system accessible to non-trusted users.

7.9 Prevent a user from killing X

To prevent a user from killing X when it is running add the following to **xorg.conf**:

```
Section "ServerFlags"
    Option "DontZap"      "True"
EndSection
```

Note

The `Ctrl+Alt+Backspace` shortcut is not directly what triggers killing the X server, but the `Terminate_Server` action from the keyboard map. This is usually not set by default, see **Xorg/Keyboard configuration#Terminating Xorg with Ctrl+Alt+Backspace**.

7.10 Killing an application visually

When an application is misbehaving or stuck, instead of using `kill` or `killall` from a terminal and having to find the process ID or name, `xorg-xkill (https://archlinux.org/packages/?name=xorg-xkill)` allows to click on said application to close its connection to the X server. Many existing applications do indeed abort when their connection to the X server is closed, but some can choose to continue.

7.11 Rootless Xorg

Xorg may run with standard user privileges instead of root (so-called "rootless" Xorg). This is a significant security improvement over running as root. Note that some popular **display managers** do not support rootless Xorg (e.g. **LightDM (https://github.com/canonical/lightdm/issues/18)** or **XDM**).

You can verify which user Xorg is running as with `ps -o user= -C Xorg`.

See also `Xorg.wrap(1) (https://man.archlinux.org/man/Xorg.wrap.1)`, `systemd-logind(8) (https://man.archlinux.org/man/systemd-logind.8)`, **Systemd/User#Xorg as a systemd user service**, **Fedora:Changes/XorgWithoutRootRights** and **FS#41257 (https://bugs.archlinux.org/task/41257)**.

7.11.1 Using xinitrc

To configure rootless Xorg using **xinitrc**:

- Run startx as a subprocess of the login shell; run `startx` directly and do not use `exec startx`.
- Ensure that Xorg uses virtual terminal for which permissions were set, i.e. passed by logind in `$XDG_VTNR` via **.xserverrc**.
- If using certain proprietary display drivers, **kernel mode setting** **auto-detection (https://gitlab.freedesktop.org/xorg/xserver/-/blob/master/hw/xfree86/xorg-wrapper.c#L222)** will fail. In such cases, you must set `needs_root_rights = no` in `/etc/X11/Xwrapper.config`.

Note that executing `startx` directly without `exec` leaves the shell open in the case of a xorg crash. Since some lock screens are executed inside xorg, this can lead to full access to the executing user.

7.11.2 Using GDM

GDM will run Xorg without root privileges by default when **kernel mode setting** is used.

7.11.3 Session log redirection

When Xorg is run in rootless mode, Xorg logs are saved to `~/.local/share/xorg/Xorg.log`. However, the stdout and stderr output from the Xorg session is not redirected to this log. To re-enable redirection, start Xorg with the `-keeptty` flag and redirect the stdout and stderr output to a file:

```
startx -- -keeptty >~/.xorg.log 2>&1
```

Alternatively, copy `/etc/X11/xinit/xserverrc` to `~/.xserverrc`, and append `-keeptty`. See [5] (https://bbs.archlinux.org/viewtopic.php?pid=1446402#p1446402).

7.12 Xorg as Root

As explained above, there are circumstances in which rootless Xorg is defaulted to. If this is the case for your configuration, and you have a need to run Xorg as root, you can configure `Xorg.wrap(1)` (https://man.archlinux.org/man/Xorg.wrap.1) to require root:

Warning

Running Xorg as root poses security issues. See #Rootless Xorg for further discussion.

```
/etc/X11/Xwrapper.config
```

```
needs_root_rights = yes
```

8 Troubleshooting

8.1 General

If a problem occurs, view the log stored in either `/var/log/` or, for the rootless X default since v1.16, in `~/.local/share/xorg/`. GDM users should check the systemd journal. [6] (https://bbs.archlinux.org/viewtopic.php?id=184639)

The logfiles are of the form `Xorg.n.log` with `n` being the display number. For a single user machine with default configuration the applicable log is frequently `Xorg.0.log`, but otherwise it may vary. To make sure to pick the right file it may help to look at the timestamp of the X server session start and from which console it was started. For example:

```
$ grep -e Log -e tty Xorg.0.log
```

```
[    40.623] (==) Log file: "/home/archuser/.local/share/xorg/Xorg.0.log", Time: Thu Aug 28 12:36:44 2014
[    40.704] (--) controlling tty is VT number 1, auto-enabling KeepTty
```

Tip

To monitor the log with human-readable timestamps, `tail(1) (https://man.archlinux.org/man/tail.1)`'s output can be piped to `ts(1) (https://man.archlinux.org/man/ts.1)` (provided by the `moreutils (https://archlinux.org/packages/?name=moreutils)` package). This will give correct timestamps only for lines added to the log while the command is running. For example:

```
$ tail -f ~/.local/share/xorg/Xorg.0.log | ts
```

- In the logfile then be on the lookout for any lines beginning with `(EE)`, which represent errors, and also `(WW)`, which are warnings that could indicate other issues.
- If there is an *empty* `.xinitrc` file in your `$HOME`, either delete or edit it in order for X to start properly. If you do not do this X will show a blank screen with what appears to be no errors in your `Xorg.0.log`. Simply deleting it will get it running with a default X environment.
- If the screen goes black, you may still attempt to switch to a different virtual console (e.g. `Ctrl+Alt+F6`), and blindly log in as root. You can do this by typing `root` (press `Enter` after typing it) and entering the root password (again, press `Enter` after typing it).

  You may also attempt to kill the X server with:

  ```
  # pkill -x X
  ```

  If this does not work, reboot blindly with:

  ```
  # reboot
  ```

- Check specific pages in **Category:Input devices** if you have issues with keyboard, mouse, touchpad etc.
- Search for common problems in **AMDGPU**, **Intel** and **NVIDIA** articles.

8.2 Black screen, No protocol specified, Resource temporarily unavailable for all or some users

X creates configuration and temporary files in current user's home directory. Make sure there is free disk space available on the partition your home directory resides in. Unfortunately, X server does not provide any more obvious information about lack of disk space in this case.

8.3 DRI with Matrox cards stopped working

If you use a Matrox card and DRI stopped working after upgrading to Xorg, try adding the line:

```
Option "OldDmaInit" "On"
```

to the `Device` section that references the video card in `xorg.conf`.

8.4 Frame-buffer mode problems

X fails to start with the following log messages:

```
(WW) Falling back to old probe method for fbdev
(II) Loading sub module "fbdevhw"
(II) LoadModule: "fbdevhw"
(II) Loading /usr/lib/xorg/modules/linux//libfbdevhw.so
(II) Module fbdevhw: vendor="X.Org Foundation"
       compiled for 1.6.1, module version=0.0.2
       ABI class: X.Org Video Driver, version 5.0
(II) FBDEV(1): using default device

Fatal server error:
Cannot run in framebuffer mode. Please specify busIDs for all framebuffer devices
```

To correct, **uninstall** the `xf86-video-fbdev (https://archlinux.org/packages/?name=xf86-video-fbdev)` package.

8.5 Program requests "font '(null)'"

Error message: `unable to load font `(null)'.`

Some programs only work with bitmap fonts. Two major packages with bitmap fonts are available, `xorg-fonts-75dpi (https://archlinux.org/packages/?name=xorg-fonts-75dpi)` and `xorg-fonts-100dpi (https://archlinux.org/packages/?name=xorg-fonts-100dpi)`. You do not need both; one should be enough. To find out which one would be better in your case, try `xdpyinfo` from `xorg-xdpyinfo (https://archlinux.org/packages/?name=xorg-xdpyinfo)`, like this:

```
$ xdpyinfo | grep resolution
```

and use what is closer to the shown value.

8.6 Recovery: disabling Xorg before GUI login

If Xorg is set to boot up automatically and for some reason you need to prevent it from starting up before the login/display manager appears (if the system is wrongly configured and Xorg does not recognize your mouse or keyboard input, for instance), you can accomplish this task with two methods.

- Change default target to `rescue.target`. See **systemd#Change default target to boot into**.
- If you have not only a faulty system that makes Xorg unusable, but you have also set the GRUB menu wait time to zero, or cannot otherwise use GRUB to prevent Xorg from booting, you can use the Arch Linux live CD. Follow the **installation guide** about how to mount and chroot into the installed Arch Linux. Alternatively try to switch into another **tty** with `Ctrl+Alt` + function key (usually from `F1` to `F7` depending on which is not used by X), login as root and follow steps below.

Depending on setup, you will need to do one or more of these steps:

- **Disable** the **display manager**.
- Disable the **automatic start of X**.

- Rename the `~/.xinitrc` or comment out the `exec` line in it.

8.7 X clients started with "su" fail

If you are getting `Client is not authorized to connect to server`, try adding the line:

```
session        optional        pam_xauth.so
```

to `/etc/pam.d/su` and `/etc/pam.d/su-l`. `pam_xauth` will then properly set environment variables and handle `xauth` keys.

8.8 X failed to start: Keyboard initialization failed

If the filesystem (specifically `/tmp`) is full, `startx` will fail. The log file will contain:

```
(EE) Error compiling keymap (server-0)
(EE) XKB: Could not compile keymap
(EE) XKB: Failed to load keymap. Loading default keymap instead.
(EE) Error compiling keymap (server-0)
(EE) XKB: Could not compile keymap
XKB: Failed to compile keymap
Keyboard initialization failed. This could be a missing or incorrect setup of xkeyboard-config.
Fatal server error:
Failed to activate core devices.
...
```

Make some free space on the relevant filesystem and X will start.

8.9 A green screen whenever trying to watch a video

Your color depth is set wrong. It may need to be 24 instead of 16, for example.

8.10 SocketCreateListener error

If X terminates with error message `SocketCreateListener() failed`, you may need to delete socket files in `/tmp/.X11-unix`. This may happen if you have previously run Xorg as root (e.g. to generate an `xorg.conf`).

8.11 Invalid MIT-MAGIC-COOKIE-1 key when trying to run a program as root

That error means that only the current user has access to the X server. The solution is to give access to root:

```
$ xhost +si:localuser:root
```

That line can also be used to give access to X to a different user than root.

9 See also

- **Xplain (https://magcius.github.io/xplain/article/)** - In-depth explanation of the X Window System
- `Xorg(1) (https://man.archlinux.org/man/Xorg.1)`
- **Prepare for LPIC-1 exam 2 - topic 106.1: X11 (https://developer.ibm.com/tutorials/l-lpic1-106-1/)** - briefly covers architecture, **#Configuration**, **desktop environments**, remote usage, **Wayland**.
- `xorg.conf(5) (https://man.archlinux.org/man/xorg.conf.5)`
- **Gentoo:Xorg/Guide#Configuration**

Retrieved from "https://wiki.archlinux.org/index.php?title=Xorg&oldid=846338"