

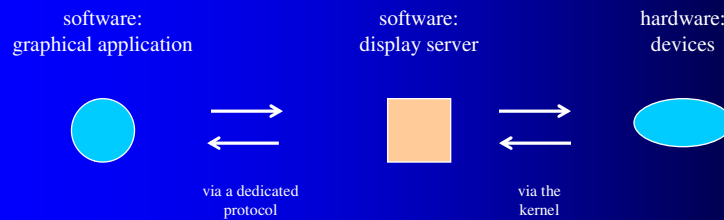
Linux Graphical Interface

David Morgan

GUI systems - a client/server model

- 1 set of devices
 - keyboard
 - mouse
 - screen
- 2 pieces of software (at least)
 - a gui client (regular graphical app)
 - wants to interact with devices
 - a display server
 - interacts with devices
- client deals with devices through server

GUI systems - a client/server model

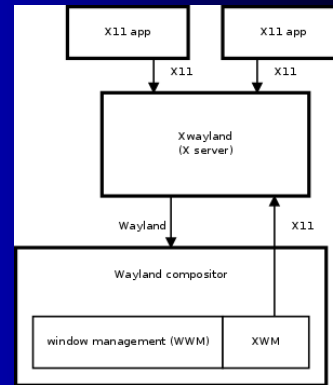


Two prevalent systems

- X window system
 - a little too old and crusty
 - originated 1984
- wayland
 - a little too young and immature
 - originated 2008

Two prevalent systems

- they can coexist in recent distributions
- Xwayland - facilitates coexistence
 - X server and wayland client
 - X clients can talk X protocol to Xwayland
 - Xwayland in turn can convert to wayland protocol



<https://wayland.freedesktop.org/docs/html/ch05.html>

The X Window System

- de facto GUI standard Unix-wide
- original project at MIT and DEC (1984)
- Version 11 Release 6 (X11R6) released 1994
- oversight and maintenance since 2004
 - x.org, an industry consortium

<http://X.org/>

X Window System topics

- Composition and architecture
- Ways to launch X
- X client-server model and networks

X Window System's pieces

- an X server (aka "X")
- X clients
 - Graphical apps
 - eg, xclock, xeyes, xterm
 - Window managers
 - eg, mwm, twm, fvwm, wmaker, sawfish, metacity
 - and more: <http://xwinman.org>
 - Desktop environments
 - eg, gnome, kde

X server's job

- track input from input devices
 - convey to relevant client apps, if any
- track output from client apps, if any
 - convey to display device
- absent: the concept of windows

X clients' function

- accept input from server
 - originated from input devices there
- send output to server
 - to be forwarded onto display there
- output is a response to input, as client desires
 - input – server to client:
“I want to let you know there was a click, at (x,y)”
 - client figures out:
hey, this click is strategic! it was on “minimize”
 - output – client to server:
“in that case please minimize that window for me”

Client example: xeyes



- X tells xeyes where cursor is, whenever it moves
- xeyes calculates corresponding pupil positions
- xeyes tells X where to put pupils
- X does so

X client-server model

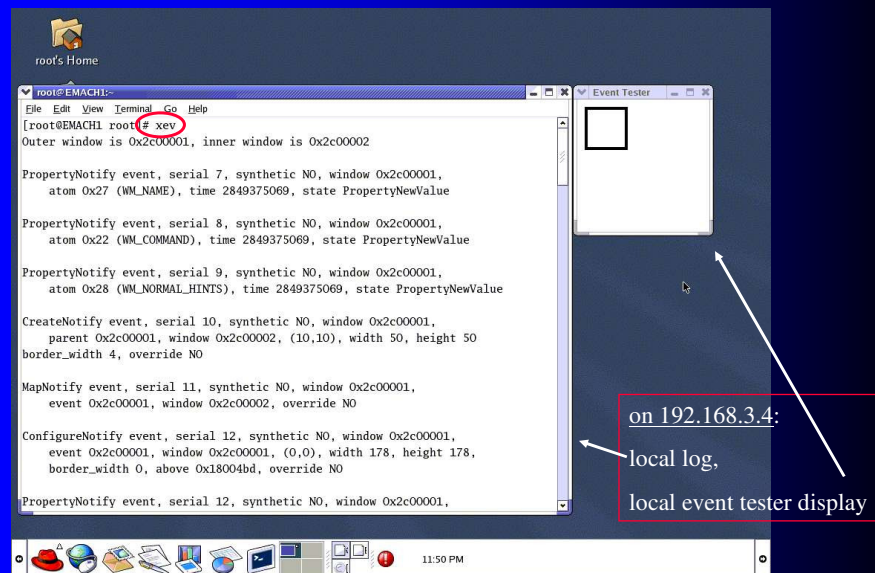
- client and server loosely coupled
- just need to communicate events (server-to-client) and requests (client-to-server)
- can be on
 - the same machine
 - different, if transport available for events/requests

server <-> client communication

User input passes from server to client in the form of events.
An event is a packet of information that tells the client something it needs to act on, such as keyboard or mouse input.
When a client program receives some kind of event, it responds with some sort of action affecting the display. For instance, it may request that a window be resized to particular dimensions. The server responds to requests by updating the appropriate window on the physical display.

X Window System User's Guide, O'Reilly, pp 21, 312
close paraphrase

xev to show local events



xev to show remote events

```
command was: xev -display 192.168.3.2:0
```

on 192.168.3.4:

local log,

event tester display on
192.168.3.2 (server)

remote activities:

left mouse button
pushed, then released

Note: modern distributions usually disable the X server's port 6000 listening feature. To demonstrate remote display you can terminate the X server and run as:

```
# xinit /usr/bin/xterm -- /usr/bin/X -listen tcp
```

Or edit startx line "listenarg="-nolisten tcp" replacing -nolisten with -listen

transport from server to client

event transport
concurrent to
button push

X11 has its own protocol – port 6000

← expresses the particular event(s) that happened

Window managers

- X clients that control other X clients
- In terms of
 - framing
 - position
 - size
 - movement
 - focus

Desktop environments

- orchestrate inter-client communication
- integrate clients/apps into common environment with consistent capabilities
 - drag and drop
 - common clipboard
 - common menus
 - uniform iconic interface to filesystem

To boot up into X

Manipulate this symbolic link (shortcut) :

`/etc/systemd/system/default.target`

So that it points to:

`/lib/systemd/system/graphical.target` (if you *do* want GUI)

or

`/lib/systemd/system/multi-user.target` (if you *don't* want GUI)

by executing:

`ln -fs /lib/systemd/system/graphical.target /etc/systemd/system/graphical.target`

or

`ln -fs /lib/systemd/system/graphical.target /etc/systemd/system/default.target`

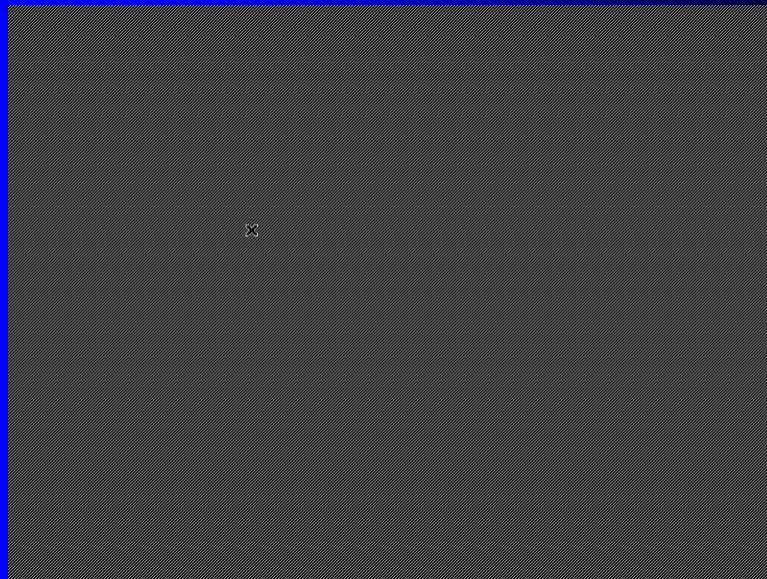
To launch X from command line

- technical but un-useful
 - X
- more useful
 - xinit
- normal
 - startx, frontend to
 - xinit, frontend to
 - xinitrc, frontend to
 - \$HOME/.Xclients, \$HOME/.Xclients-default

Using “X” to launch

- runs the X server but no client to go with it
 - X tracks input on behalf of clients but there are none
 - X sends to display any output from clients but there is none
- serves no purpose

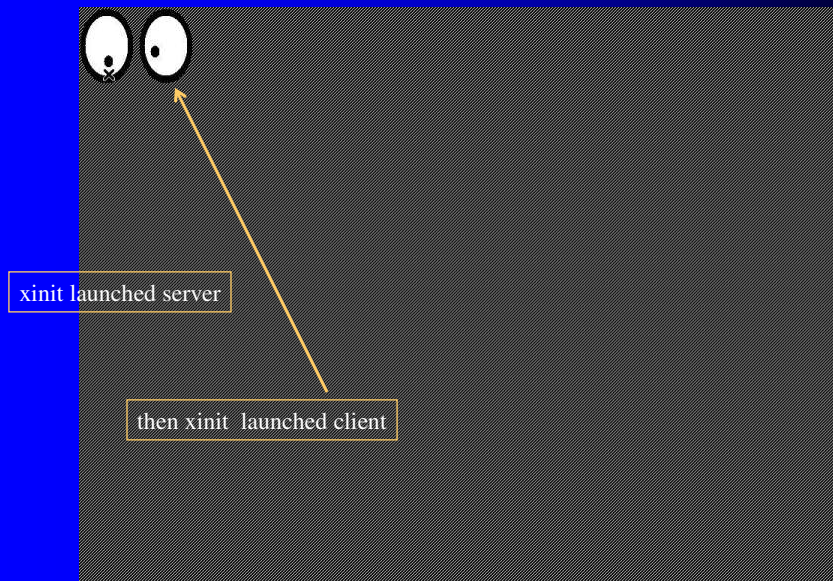
Clientless (useless) X



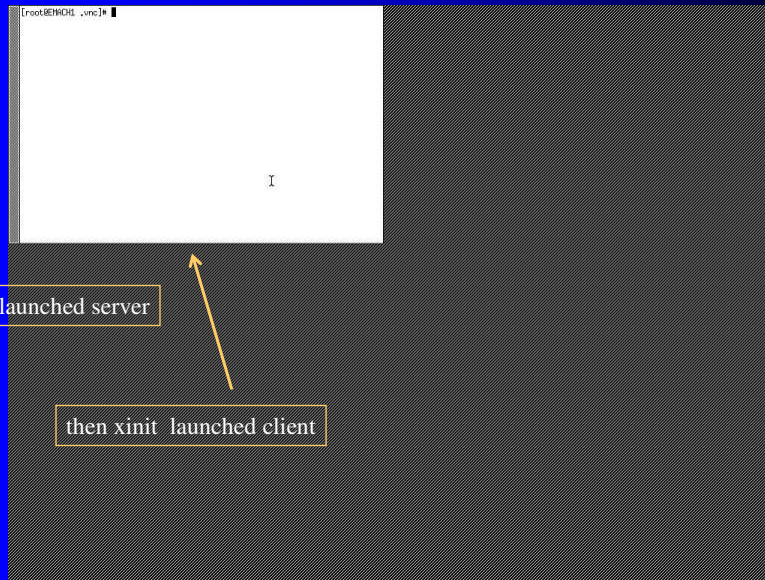
Using “xinit” to launch

- launches X server, then a client
`xinit <client> -- <server>`
- certain clients can launch further clients
 - xterms
 - window managers
- runs xterm if no <client> specified
- X server torn down when xinit’s <client> terminates

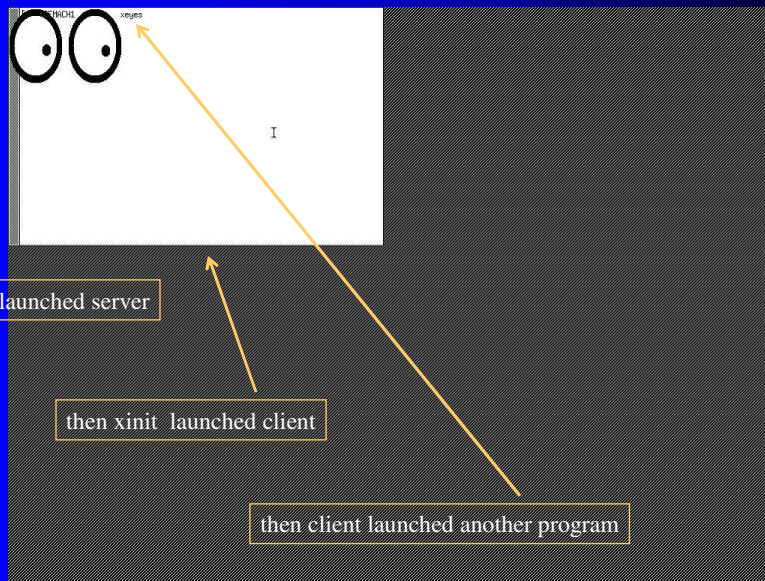
Launching “xinit”, xeyes as client



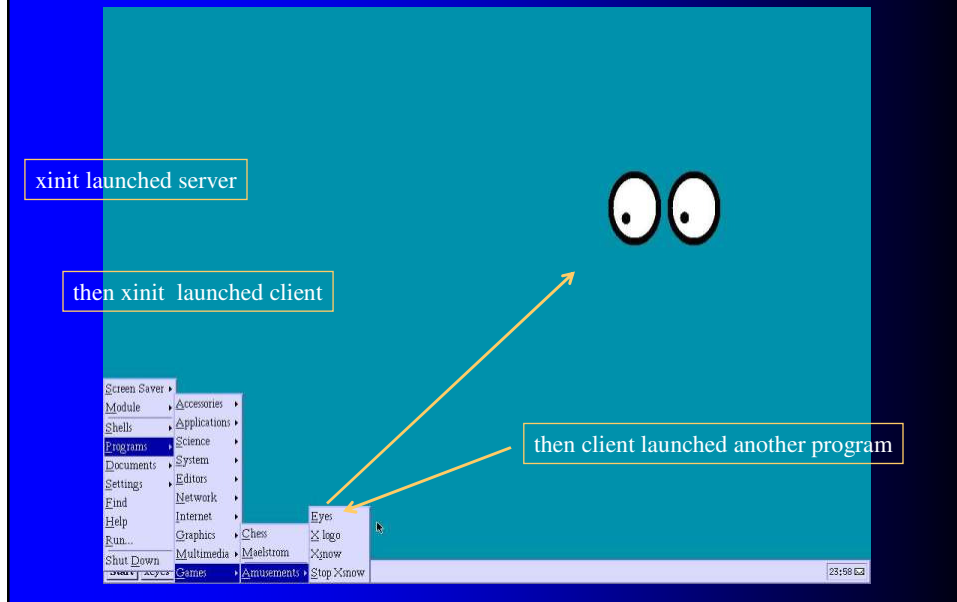
Launching “xinit”, xterm as client



...then launch xeyes from xterm



Launching “xinit”, fvwm as client



Using “startx” to launch

- starts xinit with reasonable options
- xinit after starting X, delegates client launching to a script, if any, named `.xinitrc` in user's home directory

Using “startx” to launch

- .xinitrc (if any) is your list of clients to launch

```
#!/bin/bash
# sample/simple .xinitrc shell script

#put up a clock
xclock &

#start a window manager
exec fvwm
```

- or calls it

```
#!/bin/bash
# sample .xinitrc file

exec $HOME/.Xclients
```

```
#!/bin/bash
# sample .Xclients or
# .Xclients-default file

#start a window manager
exec fvwm
```



switchdesk, auto-editor for .Xclients-default

switchdesk kde



.Xclients-default:

```
# Created by Red Hat Desktop Switcher
exec startkde
```

switchdesk gnome



.Xclients-default:

```
# Created by Red Hat Desktop Switcher
exec gnome-session
```

Some window managers

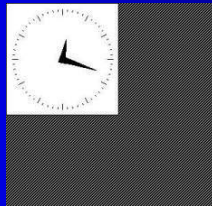
- window managers
 - mwm /usr/bin/mwm
 - twm /usr/bin/twm
 - fvwm /usr/bin/fvwm
- desktop environments
 - gnome /usr/bin/gnome-session

app, manager interfaces separate

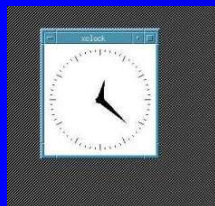
windows themselves,
controlled by managers,
differ

window content,
controlled by app,
does not

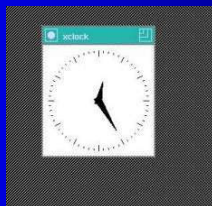
xclock with no window manager



xclock with mwm
window manager



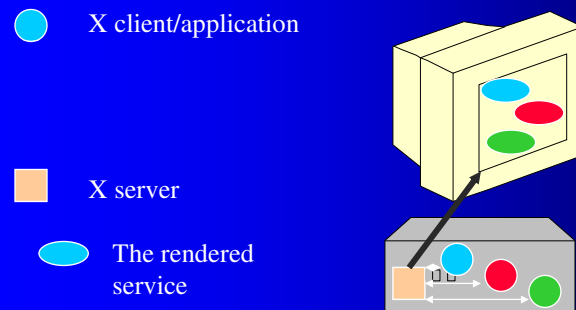
xclock with twm
window manager



xclock with window
maker window manager

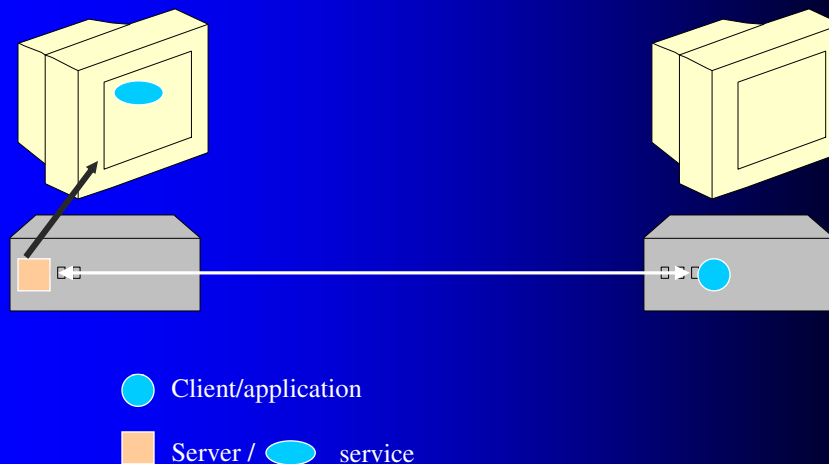


X Windows client/server model



The service is the rendered display.
The client is the application which has output to be displayed.

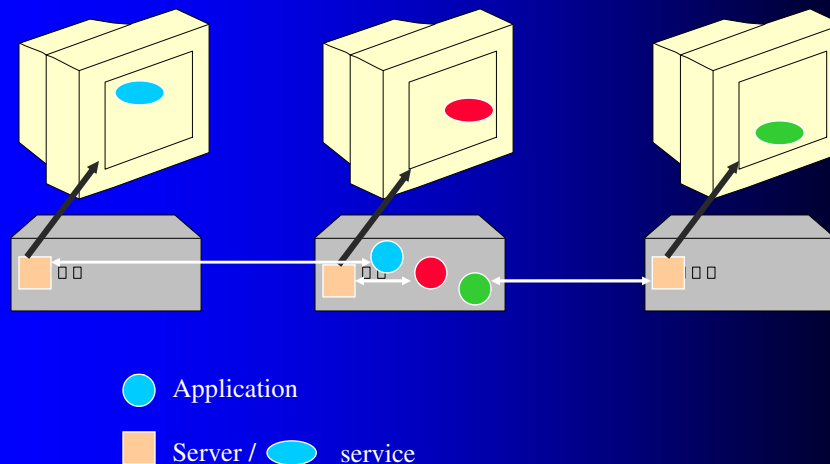
Transparently net-distributable



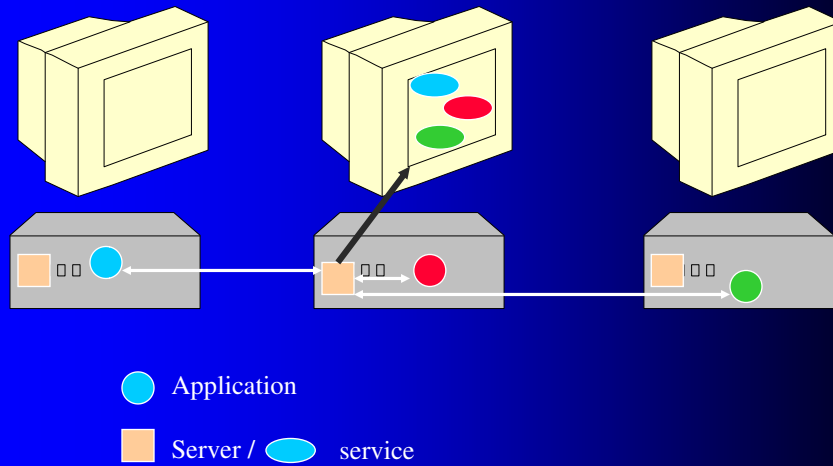
X's client-server model is counter-intuitive

- X is a *display* server, not *application* server
- application servers
 - *application is served* to the user
 - server is where the application is, client is where user sees it
- display servers:
 - *screen-rendering is served* to the application
 - client is where the application is, server is where user sees it

Apps on a machine, screen output elsewhere



Screen output on a machine, apps elsewhere



Operation

Issued in the client, where xapp resides:

```
$ xapp -display <server>:0
```

Prerequisite server permission, in server:

```
$ xhost +
```

Special-guest client appearance!

- any client can display to a server elsewhere
- window managers are clients
- so a window manager can display to a server elsewhere (thus create and manage its windows)
- subject to the not-more-than-one window manager limitation on any server
 - launch X on server using `xinit` not `startx` (to avoid a wm)
 - then apply/display a wm as client from another machine (e.g., run “`fvwm –display <this box>:0`” on that box)

Wayland

- does the job differently
 - has a different architecture
 - uses a different protocol

Wayland architecture

- performance - involves fewer processes
 - therefore fewer context switches
- security - not centralized
 - all activity runs through an X11 server
 - one-stop shopping for malware
 - client activity in Wayland is individual/isolated

Wayland client programs

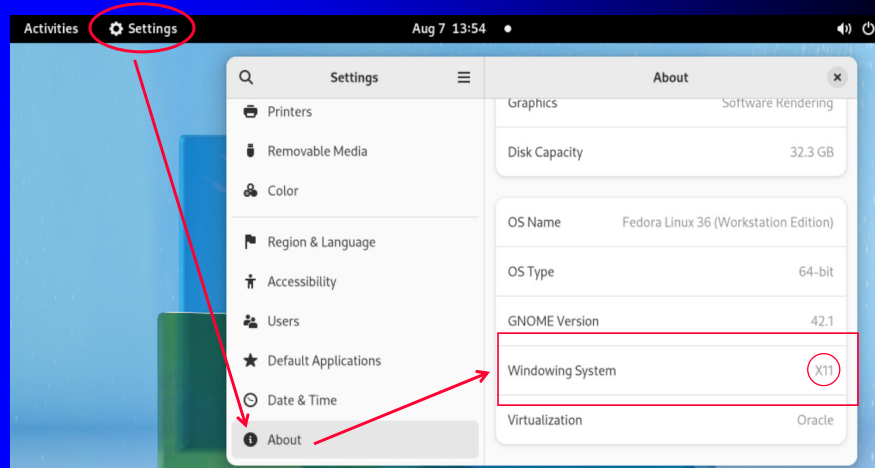
- programs must be coded accordingly
 - new programs
 - written from scratch to use wayland protocol
 - legacy X11 programs
 - ported to speak wayland protocol instead or in addition to X11

Two "bilingual" programs

- Gnome desktop environment
- Firefox web browser

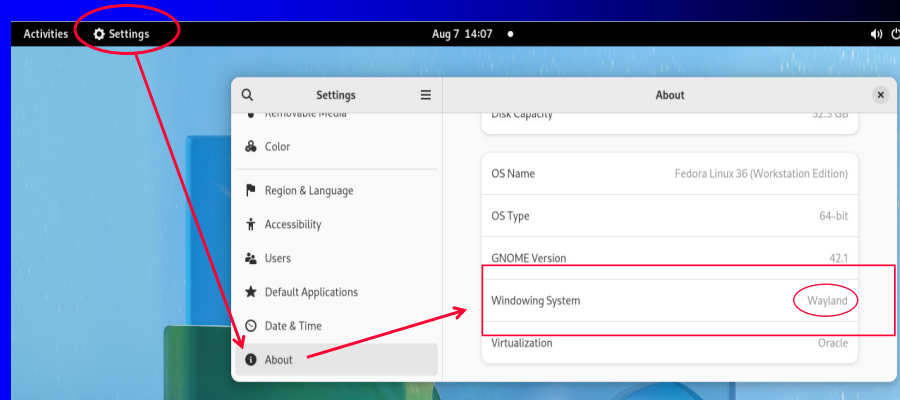
Gnome

launched by "startx"



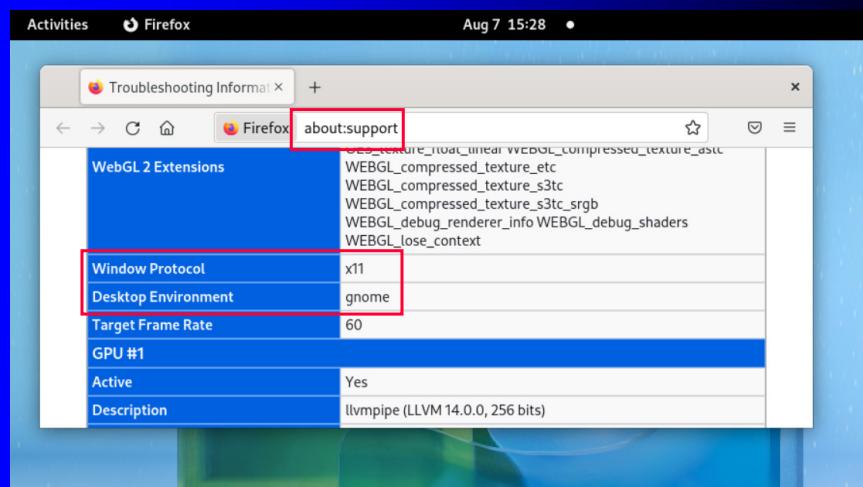
Gnome

launched by "dbus-run-session -- gnome-shell --display-server --wayland"



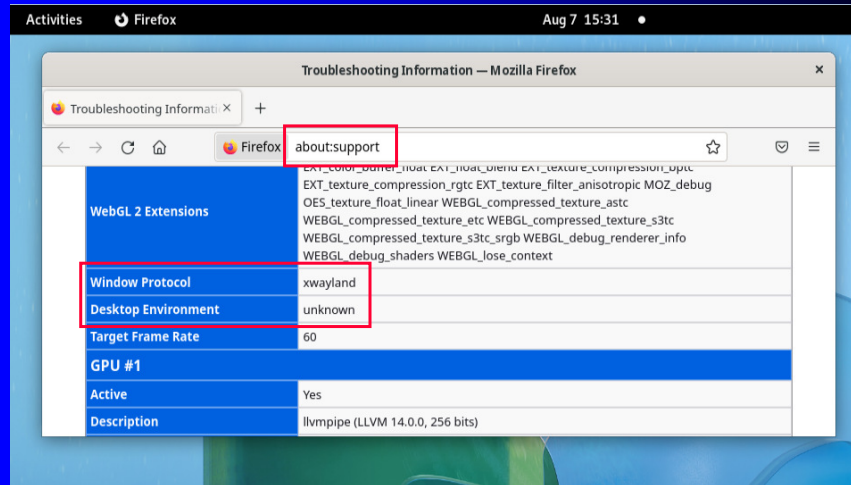
Firefox

launched by icon



Firefox

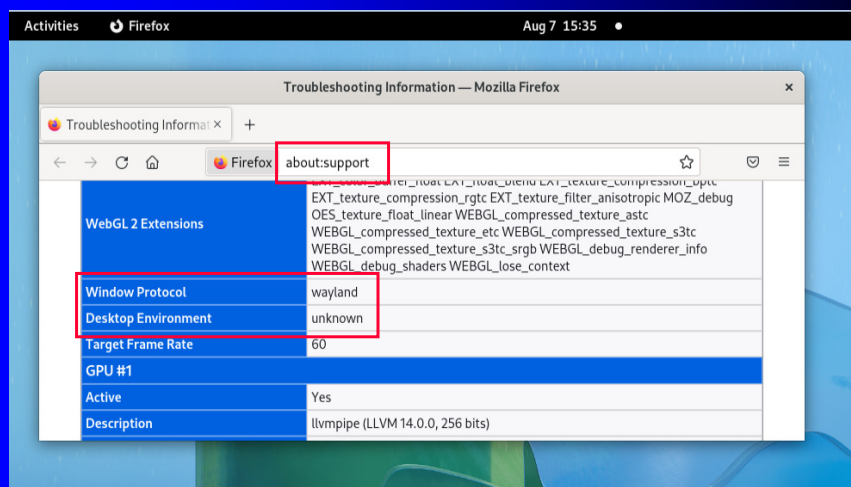
launched by shell command "firefox"



Firefox

launched by shell command

with MOZ_ENABLE_WAYLAND
environment variable set to 1
by "export MOZ_ENABLE_WAYLAND=1" command



Some resources

- <https://wayland.freedesktop.org/>
- <https://linuxiac.com/xorg-x11-wayland-linux-display-servers-and-protocols-explained/>
- https://www.slant.co/versus/8634/8635/~wayland_vs_x-org