

# LPIC-1 102-500 – Lesson 3

## 106.1 Install and configure X11



# Terminology

- **Desktop Environment:** A collection of programs, icons, themes to provide a unique user interface. They use **Window Managers** to talk to the **X Window System (X Server)**.
- **Window Manager:** a system that generates Windows (or Frames). It serves as a client to the **X Window System**.
- **X Window System:** it the server that provides the underlying system for the Graphical environment to work.



# Window Managers

- **Mutter:** used in GNOME
- **KWin:** used in KDE
- **Compiz:** A popular Window Manager with 3D effects
- **AwesomeWM:** A tiling Window Manger



# The X Window System (X11)

- The **X Window System** (X11 or simply **X**) is a windowing system traditionally used on Graphical Linux Installations
- **X.Org** (Xorg) is the most popular implementation of the X11 standard. Other implementations are **XFree86** and **Xwin** for MS Windows.
- It uses the *client-server model*. That means it has to be paired with a *Window Manager* (client). By default that Window Manager is **TWM**.
- Because of various design limitations it is gradually being replaced by the **Wayland** display server protocol

# Installing X.Org

- Installing Xorg on Debian/Ubuntu:

```
$ sudo apt install xorg # Full
```


```
$ sudo apt install xserver-xorg-core # Server only
```

- Installing Xorg on RedHat/CentOS/Fedora:

```
$ sudo yum groupinstall "X Window System" # replace  
yum with dnf of Fedora and recent versions of RedHat/  
CentOS
```



# Starting X

- To start the Graphical Environment on a CLI-only Linux system, we have to run **X** or **startx**
  - **X** is usually a symlink to the **Xorg** executable. We usually avoid running **X** directly. Running **startx** is preferable.
  - **startx** is a script that prepares the environment to allow **X** to start.
  - If **X** fails to start a *.xsession-errors* file will be created under your home directory.
- 



# Display Managers

**X** is rarely started directly from the command line. Usually we use services called **Display Managers** that handle the session and the graphical login. Some Display Managers:

- **Gdm**: Used by the Gnome Desktop
- **Kdm**: Used by the KDE Desktop
- **Xdm**: A simple Display Manager for X
- **LightDM**: A lightweight Display Manager



# The Xorg configuration

- Usually under */etc/X11/xorg.conf*
- You can also create your own custom configuration files under */etc/X11/xorg.conf.d* (ending in *.conf*)
- A new sample *xorg.conf* can be generated on a Gui-less terminal (ex. Ctrl-Alt-F2) with:

```
$ sudo X :1 -configure
```

```
$ sudo ls -l /root/xorg.conf.new
```

```
-rw-r--r-- 1 root root 3306 Nov 26 10:08 /root/xorg.conf.new
```





# Xorg Configuration Sections

- **Section "Files"**: Files (usually fonts) to load
- **Section "Module"**: Modules to be loaded
- **Section "InputDevice"**: Input devices like keyboard/Mouse
- **Section "Device"**: Graphics card
- **Section "Monitor"**: Monitor Definition
- **Section "Screen"**: Screen definition (resolution etc)
- There are some other less definitions as well. Have a look at the *xorg.conf* file under **Lesson3**

# The *\$DISPLAY* environment variable

- The **\$DISPLAY** environment variable sets the display used by the system. It is possible to set a different display for remote access or have two displays on a system, ex. :0.0 and :0.1
- **\$ echo \$DISPLAY #** show the **\$DISPLAY** variable  
:0.0 # :0.1, :0, :1 are other possible choices
- If you want to use the display on another computer:

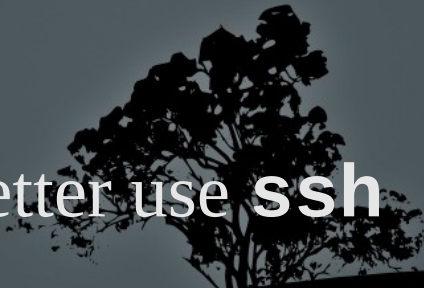
```
$ export DISPLAY=remote.cut.edu:0.0
```



# Accessing a remote X server using *xhost*

- `root@host1 # xhost +host2` # the remote host `host2` is granted the right to use the X server on `host1`
- `root@host1 # xhost +10.0.0.10` # we can also use the ip address of `host2`
- `root@host2 # export DISPLAY=host1:0.0` # we can also use the IP of `host1`, ex:  
`export DISPLAY=10.0.0.11:0.0`
- `root@host2 # gedit` # the `gedit` graphical application runs on `host2` but displayed on `host1`

**NOTE:** This method is very insecure. Better use `ssh -X` instead



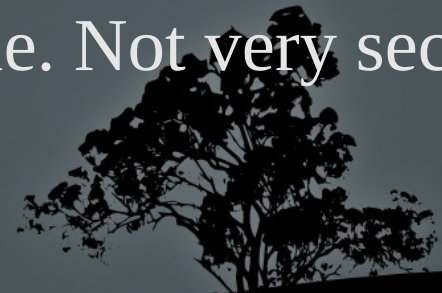
# Other authentication methods

- SSH X11 Tunneling: This is the most secure method as it uses the SSH protocol for tunneling:

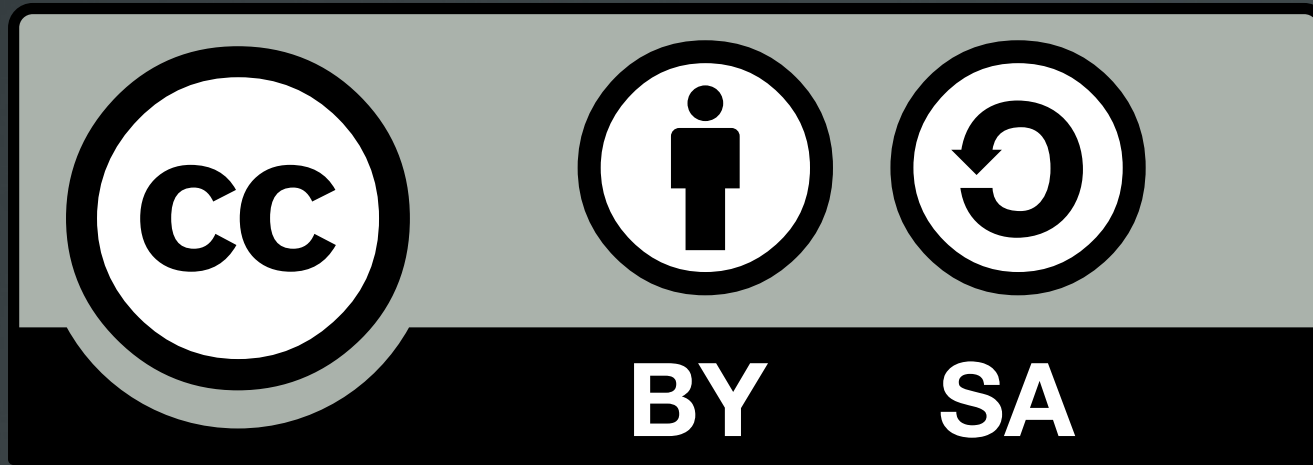
```
local~$ ssh -X user@remote.system  
remote~$ xeyes
```

In this example the **xeyes** graphical application runs on X server of the remote system but displayed on the local system.

- **xauth** is another X authentication method which adds remote hosts in the X configuration file. Not very secure



# License



The work titled "LPIC-1 102-500 – Lesson 3" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

