Xmonad Tutorial for Beginning Beginners Documentation

Release 1.0

Avery Laird

February 22, 2017

Contents

If you are wanting to get started with using Xmonad, but are new to ubuntu or unix-like systems and/or new to setting up window manager environments yourself, this is the tutorial for you. It will not go into great depth for the sake of simplicity, but I will cite some other sources where you can find the more complicated explanations.

Contents:

CHAPTER 1

Introduction to using Xmonad

The most important thing to understand about Xmonad is that it was created as a barebones, no distractions window manager. Because of this, there is a lot of customization and tweaking involved to get a working environment you are completely happy with. This may seem like a downside. However, consider the control one now has over their desktop environment – the sky's the limit. If having to get your hands dirty crafting some custom code seems like a daunting task, that's understandable. But, everyone has to start somewhere, and if you're interested in some serious desktop overhaul then here's the place to start.

I created this tutorial because I realized that there was a gap in Xmonad documentation. There existed copy-and-paste docs that explained nothing, or lengthy explanatory docs that, unforunately, didn't have anything to copy-and-paste! That's why I created the tutorial for beginning beginners – it's half explanation and half code samples to get you started. If you're ready, let's get to *Installing Xmonad*.

CHAPTER 2

Installing Xmonad

When installing xmonad, pretty much any package manager will do. You can even compile from source, but if you're reading the beginning beginner's guide to xmonad, I would recommend against that unless you follow an extremely clear guide (like this). I use APT for package management.

Install xmonad and some handy tools

Install xmonad and dmenu:

```
sudo apt-get install xmonad suckless-tools xscreensaver
```

The package `suckless-tools` installs dmenu on my latest version of ubuntu just fine. Older versions may have to install `dwm-tools` instead, and other distros might have to search for dmenu as a separate package. Dmenu is a bit of a must-have for getting started, because it allows the execution of applications easily, as we'll see later. If you decide for some reason later on that dmenu is not for you, it's easily uninstallible however I recommend that you get started with dmenu. The `xscreensaver` package will be used later on, with handling lock screens.

Configuration and tweaking (and maybe some more installing)

If you were to login using xmonad right now, you would be greeted with an entirely black screen. For the sake of simplicity, let's hold off on that just yet and do a bit more housework first. Next we're going to install xmobar, a handy status bar popular among xmonad users. Like before, I'm going to use APT to install:

```
sudo apt-get install xmobar
```

Once again, xmonad would still show us a nice black screen. Like pretty much anything in xmonad, xmobar requires some hefty configuration to start looking pretty. That configuration is supplied by a file in your home directory, `~/.xmobarrc`. You might have to create it, or it may already be there. If it's not there, do `touch ~/.xmobarrc`, and use any text editor you like to make the necessary changes. I prefer vim for small things like this. With that done, let's move on to the next section.

CHAPTER 3

Configuring Xmobar

Here's what's inside my `.xmobarrc`:

```
Config { font = "-*-Fixed-Bold-R-Normal-*-13-*-*-*-*-*-*-*"
        , borderColor = "black"
        , border = TopB
        , bgColor = "black"
        , fgColor = "grey"
        , position = TopW L 100
        , commands = [ Run Weather "CYVR" ["-t","<tempC>C","-L","18","-H","25","--normal","green","--
                        , Run Network "eth0" ["-L","0","-H","32","--normal","green","--high","red"] 1
                        , Run Network "eth1" ["-L","0","-H","32","--normal","green","--high","red"] 1
                        , Run Cpu ["-L","3","-H","50","--normal","green","--high","red"] 10
                        , Run Memory ["-t","Mem: <usedratio>%"] 10
                        , Run Swap [] 10
                        , Run Com "uname" ["-s","-r"] "" 36000
                        , Run Date "%a %b %_d %Y %H:%M:%S" "date" 10
                        , Run StdinReader
                        ]
        , sepChar = "%"
        , alignSep = "}{"
        , template = "%StdinReader% | %cpu% | %memory% * %swap% | %eth0% - %eth1% }{<fc=#ee9a00>%date
        }
```

Whoa, that's a lot of options! Let's break it down line by line:

`, borderColor = "black"`: Defines xmobar's border colour as black.

`, border = TopB`: Draws a border at the top of xmobar's window. Black in this case, as defined in the previous setting.

`, bgColor = "black"`: Defines xmobar's background colour as black.

`, fgColor = "grey"`: Defines the default font colour as grey.

`, position = TopW L 100`: Defines the position of xmobar as the top of the screen, on the left hand side, taking up 100% of width of the screen.

`, commands = []`: This defines a list of commands, or things to do when xmobar starts up. All of the newlines starting with a comma and `Run` define different ways of getting information from different areas of the system, and how to display that information on the bar.

`, sepChar = "%"`: Character to be used for indicating commands in the output template. (`%` is the default anyways).

`, alignSep = "}{"`: The characters that define which commands in the output template are aligned left, and which are aligned right. Any commands on the left side of a `}` are aligned to the left side, and any on the right side of a `{` are aligned to the right side. This will make a bit more sense in the next (and final) explanation.

`, template = "%StdinReader% | %cpu% | %memory% * %swap% }{<fc=#ee9a00>%date%</fc>| %eth0% - %eth1% | %uname% | %CYVR% "`: This one may not be immediately obvious at first glance. Basically, this is a template to define how the data retrieved by the various commands we ran earlier should be displayed. Going from left to right, it says that the data from `StdinReader`, CPU usage, memory usage and swap usage should be shown on the left. Now we see here our friends the curly braces; as I explained before, they say that all that data should be on the left hand side. On the right hand side is the date (in orange, as defined by the `<fc>` tags), the download and upload speeds, kernel name and the temperature. I live in Vancouver, hence `CYVR`. *You'll have to change that value to your area code.*

Final thoughts

You may not understand completely how the configuration file works at first, however give it time and slowly it will start to make sense. As we continue to delve deeper into the xmonad universe, the various connections between files will become more and more easy to follow. I promise! On we go to configuring stalonetray.

CHAPTER 4

Configuring Stalonetray

Stalonetray is a small “stand-alone” tray, nice easy and simple to use. It’s installed with a simple APT command: `sudo apt-get install stalonetray`. Like xmobar, it also gets it’s parameters from a file in your home directory called `~/.stalonetrayrc`. Also like xmobar, if the file does not yet exist, create it.

Here’s what my stalonetray configuration looks like:

```
decorations none
transparent false
dockapp_mode none
geometry 5x1-400+0
max_geometry 5x1-325-10
background "#000000"
kludges force_icons_size
grow_gravity NE
icon_gravity NE
icon_size 12
sticky true
#window_strut none
window_type dock
window_layer bottom
#no_shrink false
skip_taskbar true
```

I will go into less detail for these options than xmobar, because the documentation is more or less quite easy to understand. I will touch on the more important ones, however:

`geometry 5x1-523+0`: The first two values, `5x1` are the height and width of the bar (measured in multiples of icon slots, e.g. 5 icons wide and 1 icon high). The next two are the `x` and `y` positions.

`background "#000000"`: Black background, to match xmobar.

What Next

If you were to run stalonetray right now, most likely it would simply be an empty black bar. With no icons to fill it, it would just blend right in with xmobar. In the next part, however, when we configure the `xmonad.hs` file, processes will be started on login and will populate stalonetray. Let’s get started!

CHAPTER 5

Configuring Xmonad.hs

Now we are nearing the point where we could login to xmonad and be greeted by some friendly readouts, and perhaps a sign that the login was successful. As everything stands now:

- xmonad and dmenu are installed
- xmobar is installed and configured
- stalonetray is installed and configured

Next, we have to configure xmonad itself. As you might have noticed, we're following a bit of a pattern: install the software, configure the software and etc. After that, we tie it all together in the `~/.xsessonrc` file which is where all of the software is started during login. Once they're started up, the software then references their config files.

Xmonad file structure

The configuration files for xmonad are layed out a little bit like this:

```
~/.xmonad
     |
     +-- xmonad.hs
```

Now, a lot of the time (or at least some of the time) you have to create this file structure yourself. This is quite simple.

```
sudo mkdir ~/.xmonad && cd ~/.xmonad
```

And then:

```
touch xmonad.hs
```

Or if you're like me and like to skip steps (and have vim installed) then you can do:

```
vim xmonad.hs
```

Notice the period before `.xmonad` and lack of period before `xmonad.hs`. This is important to get right.

Configure Xmonad

`Xmonad.hs` is written in Haskell (as is the rest of xmonad itself) and can contain pretty any configuration you like. Mine is rather simple, and is a mish mash of various things I've found on the web and various things I've done myself:

```
import XMonad
import XMonad.Hooks.DynamicLog
import XMonad.Hooks.ManageDocks
import XMonad.Util.Run(spawnPipe)
import XMonad.Util.EZConfig(additionalKeys)
import System.IO

main = do
    xmproc <- spawnPipe "xmobar"

    xmonad $ defaultConfig
        { manageHook = manageDocks <+> manageHook defaultConfig
        , layoutHook = avoidStruts  $  layoutHook defaultConfig
        , logHook = dynamicLogWithPP xmobarPP
                        { ppOutput = hPutStrLn xmproc
                        , ppTitle = xmobarColor "green" "" . shorten 50
                        }
        , modMask = mod4Mask     -- Rebind Mod to the Windows key
        } `additionalKeys`
        [ ((mod4Mask .|. shiftMask, xK_z), spawn "xscreensaver-command -lock; xset dpms force off")
        , ((controlMask, xK_Print), spawn "sleep 0.2; scrot -s")
        , ((0, xK_Print), spawn "scrot")
        ]
```

I won't go into large detail like I have with the other files, mostly because this isn't so much a configuration file as an entity of xmonad itself and also because the documentation on Haskell (and xmonad with Haskell) are quite extensive. If you're interested, here is a good place to start.

The most notable things about this file you need to now is that the primary xmonad key (which you may not know, but so far has been Mod/ALT) is remapped to the Meta, or Windows, key. The last two lines are for taking screen shots (which I barely use) but one useful keybinding (`Shift + Meta + Z`) is for locking the screen. It invokes `xscreensaver-command -lock` and then `xset dpms force off` to actually switch off the monitor (I've found that my laptop backlight stays on, and in an effort to save power and pixels I have included this command as well). For extensive config examples and inspiration, try looking on the web – or try something like this.

CHAPTER 6

The `.xsessionrc` File

Here we are, the (kind of) final step! After this, you should have a basic, working instance of xmonad. The `.xsessionrc` file basically sets up all the instances needed (xmobar, stalonetray, etc) at login. It's simply a bash script, with the whole "shebang" so to speak! (I couldn't resist, look here if you're a little confused)

Anyways, bad jokes aside, here's the joe-basic script that'll get you going for now. It will *have* to be placed in `~/.xsessionrc` (you can do `touch ~/.xsessionrc` or like I also mentioned, `vim ~/.xsessionrc`:

```
#!/bin/bash

# Load resources

xrdb -merge .Xresources

# Set up an icon tray
stalonetray &

# Fire up apps

xscreensaver -no-splash &

if [ -x /usr/bin/nm-applet ] ; then
   nm-applet --sm-disable &
fi

exec xmonad
```

Now, this is very simple: it starts stalonetray, the xscreensaver daemon, the wifi applet, and then does some magic with `stdout` and xmonad (look here if you're interested).

Congratulations! You've taken the first big step in the world of DIY window managing.

Further Tweaks

Now, at this point, many simple things about xmonad are working, like:

- xmonad itself
- xmobar
- stalonetray (your icon bar)
- xscreensaver

However, as I said before, xmonad doesn't assume anything – meaning, there are many things you may still want working that xmonad does not do automatically like gnome or xfce. I chose to end part of the tutorial here for people who don't care about those things yet, but if you want the following, try out the next section:

- Wallpaper
- Automatic handling of external media (mounting cd's, usb drives, etc)
- Power management (battery info, suspension, etc)

CHAPTER 7

Wallpaper

This is really simple and easy to do, and there are a huge number of ways to do it. This is my method, using `feh`.

1. First, install `feh`:

```
sudo apt-get install feh
```

2. Second, configure the wallpaper to be set on startup. For this, we'll need the `~/.xsessionrc` file. Place this line somewhere in `.xsessionrc`, after `#!/bin/bash` but before the `if` statements:

```
# Set background image with feh
feh --bg-scale /usr/share/backgrounds/warty-final-ubuntu.png &
```

This just invokes `feh` and tells it which image to set as the background. I have the default warty background here, but you can change the file path to any image you want.

CHAPTER 8

Handling External Media

This is also quite easy as well, thanks to a handy little python package called `udiskie`. Here are the steps:

1. Install `pip` to install `udiskie`:

```
sudo apt-get install python-pip
```

2. Install `udiskie` using `pip`:

```
sudo pip install udiskie
```

3. Start `udiskie` on login, by placing a line in `~/.xsessionrc`:

```
# Start udiskie to handle media
udiskie &
```

Now, any inserted media should be handled automatically by udiskie. There are many other ways of doing this (such as using nautilus or some other file browser) that are easily researched through google as well.

Power Management in xmonad

Okay, this is also simple as well:

1. Install `xfce4-power-manager`:

```
sudo apt-get install xfce4-power-manager
```

2. Start it on login, in the `~/.xsessionrc` file. Once again, this should be after the shebang but before the `if` statements:

```
xfce4-power-manager &
```

The power manager icon should now show up in stalonetray.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search