Xmonad as a Windowmanager for Slackware

Xmonad is a tiling window manager. For information about tiling window managers please read this wiki: wikipedia

For Xmonad read here

Required packages

Xmonad is not included in Slackware by default, but available via SlackBuilds.org. Xmonad is written in Haskell and therefore some packages of the Haskell series are required in order to build Xmonad. Here are the packages in the correct build order:

1. ghc (the glasgow-haskell-compiler)
2. haskell-syb
3. haskell-utf8-string
4. haskell-X11
5. haskell-transformers
6. haskell-mtl
7. xmonad
8. haskell-random
9. xmonad-contrib
10. haskell-hinotify
11. haskell-stm
12. haskell-X11-xft
13. haskell-text
14. haskell-parsec
15. xmobar (provides a statusbar)

I have additionally installed dmenu which is integrated into the statusbar and starts programs (like gmrun). I have also installed `trayer` which provides a systray in the statusbar. Unfortunately `trayer` is only available as an rpm-package. I wanted to write a SlackBuild script for it, but the sources are incomplete. Another tray is stalonetray which is available via SlackBuilds.org.

Configuration of Xmonad

After building and installing the above packages you can configure Xmonad. One remarkable feature of xmonad as well as xmobar is that it is not only written in the functional language Haskell, but also the configuration is a Haskell file. This makes it a bit difficult to understand the configuration files if one doesn't know Haskell. Well, I once tried to learn Haskell but (yet) without success.

At first one has to configure `.xinitrc` in order to start Xmonad correctly when changing from runlevel 3 to 4.

.xinitrc

the following sections of my `.xinitrc` configure `dbus`, the mouse pointer and `trayer`, then xmonad is started

```
# Use dbus-launch if installed.
if test x"$DBUS_SESSION_BUS_ADDRESS" = x""; then
 dbuslaunch=`which dbus-launch`
 if test x"$dbus-launch" != x"" -a x"$dbus-launch" != x"no"; then
  eval `$dbus-launch --sh-syntax --exit-with-session`
 fi
fi
xsetroot -cursor_name left_ptr

trayer --edge top --align right --SetDockType true --SetPartialStrut true \
         --expand true --width 10 --transparent true --height 14 &

exec xmonad
```

.xmobarrc

xmobar is a statusbar and displays useful information, in my case in the top part of the desktop. Below is an example of my `.xmobarrc`:

```
Config { font = "-misc-fixed-bold-R-normal-*-13-*-*-*-*-*-*-*"
    , bgColor = "#1074EA"
    , fgColor = "#DDDDDD"
    , position = TopW L 90
    , commands = [ Run BatteryP ["BAT1"]
                 ["-t", "<acstatus><watts> (<left>%)",
                  "-L", "10", "-H", "80", "-p", "3",
                  "--", "-O", "<fc=green>On</fc> - ", "-o", "",
                  "-L", "-15", "-H", "-5",
                  "-l", "red", "-m", "blue", "-h", "green"] 60
          , Run Cpu ["-L","3","-H","50","--normal","green","--high","red"]
10
          , Run CpuFreq ["-t", "<cpu0> <cpu1>", "-L", "0", "-H", "2",
                  "-l", "lightblue", "-n","white", "-h", "red"] 50
          , Run Memory ["-t","Mem: <usedratio>%"] 10
          , Run Swap [] 10
          , Run Date "%a %d. %B %H:%M Uhr" "LC_TIME=de_DE date" 10
          , Run StdinReader
          ]
    , sepChar = "%"
    , alignSep = "}{"
    , template = "%StdinReader% }{ <fc=#FFD700>%date%</fc> | %cpu%
%cpufreq% | %memory% %swap% | Bat: %battery% "
```

```
    }
```

The first lines configure the font, foreground/background-colors and the position on the screen. The rest configures the information which should be shown in xmobar, battery-state, CPU-load, CPU-frequency, Memory-usage and Swap-usage, the date. Note that `LC_TIME=de_DE date` forces the date command to use the language defined in `LC_TIME` (German in my case).

For further explanation please read the manuals.

xmonad.hs

Here is an example of my `~/.xmonad/xmonad.hs` file

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

myManageHook = composeAll
  [ className =? "Gimp"    --> doFloat
  , className =? "Vlc"     --> doFloat
  ]

main = do
  xmproc <- spawnPipe "/usr/bin/xmobar /home/markus/.xmobarrc"
  xmonad $ defaultConfig
    { manageHook = manageDocks <+> manageHook defaultConfig
    , layoutHook = avoidStruts $ layoutHook defaultConfig
    , logHook = dynamicLogWithPP xmobarPP
            { ppOutput = hPutStrLn xmproc
            , ppTitle = xmobarColor "green" "" . shorten 50
            }
    } `additionalKeys`
    [ ((mod4Mask, xK_c  ), kill)
     ,((mod4Mask, xK_Return ), spawn "xterm")
    ]                           ]
```

Please read the documentation for `xmonad.hs`. This is only an example (which works well for me).

Additional Hints

One can reload the configurations for xmobar and/or xmonad after changes with MOD+q without leaving X. This is very useful.

When using a tiling window manager one experiences that some applications behave unusual. In my `xmonad.hs` file above you see `Vlc` and `Gimp` in the list of programs which should float. In order to

find out the so called `Classname` of the application (through which the application can be detected by the window manager) there is a script in the xmonad-contrib package. You can find it in `/usr/share/doc/xmonad-contrib-0.10/scripts/` directory.

Sources

- Originally written by Markus Hutmacher

howtos, windowmanager, tiling-windowmanager, haskell, xmonad, author markush