Xmonad como un administrador de ventanas para Slackware

Xmonad es un gestor de ventanas en mosaico. Para obtener información sobre los administradores de ventanas en mosaico, lea este wiki: wikipedia

Para Xmonad lea aquí

Paquetes requeridos

Xmonad no está incluido en Slackware de manera predeterminada, pero está disponible a través de SlackBuilds.org. Xmonad está escrito en Haskell y, por lo tanto, se requieren algunos paquetes de la serie Haskell para construir Xmonad. Aquí están los paquetes en el orden de compilación correcto:

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)

Además, instale dmenu que está integrado en la barra de estado e inicia programas (como gmrun). También he instalado `trayer` que proporciona una bandeja del sistema en la barra de estado. Desafortunadamente, “trayer” solo está disponible como un paquete rpm. Quería escribir un script de SlackBuild para él, pero las fuentes están incompletas. Otra bandeja es stalonetray, que está disponible a través de SlackBuilds.org.

Configuración de Xmonad

Después de compilar e instalar los paquetes anteriores, puede configurar Xmonad. Una característica notable de xmonad y xmobar es que no solo está escrito en el lenguaje funcional Haskell, sino que también la configuración es un archivo Haskell. Esto hace que sea un poco difícil entender los archivos de configuración si uno no conoce a Haskell. Bueno, una vez intenté aprender Haskell pero (todavía) sin éxito.

Al principio, uno debe configurar `.xinitrc` para iniciar Xmonad correctamente al cambiar de nivel de ejecución 3 a 4.

.xinitrc

Las siguientes secciones de mi `.xinitrc` configuran `dbus` , el puntero del mouse y `trayer` , luego se inicia xmonad

```
# 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 es una barra de estado y muestra información útil, en mi caso en la parte superior del escritorio. A continuación se muestra un ejemplo de mi `.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% "
```

```
    }
```

Las primeras líneas configuran la fuente, los colores de primer plano/fondo y la posición en la pantalla. El resto configura la información que debe mostrarse en xmobar, estado de batería, carga de CPU, frecuencia de CPU, uso de memoria y uso de intercambio, la fecha. Tenga en cuenta que `LC_TIME = de_DE date` obliga al comando de fecha a usar el idioma definido en `LC_TIME` (alemán en mi caso).

Para más explicaciones, lea los manuales.

xmonad.hs

Aquí hay un ejemplo de mi archivo `~ / .xmonad / xmonad.hs`

```
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")
    ]                           ]
```

Lea la documentación de `xmonad.hs` . Este es solo un ejemplo (que funciona bien para mí).

Consejos adicionales

Uno puede recargar las configuraciones para xmobar y/o xmonad después de los cambios con MOD + q sin salir de X. Esto es muy útil.

Cuando se usa un administrador de ventanas de mosaico, se experimenta que algunas aplicaciones

se comportan de manera inusual. En mi archivo `xmonad.hs` arriba, verá `Vlc` y `Gimp` en la lista de programas que deberían flotar. Para descubrir el llamado `Nombre de clase` de la aplicación (a través del cual el administrador de ventanas puede detectar la aplicación) hay un script en el paquete xmonad-contrib. Puede encontrarlo en el directorio `/usr/share/doc/xmonad-contrib-0.10/scripts/` .

Funetes

- Originalmente escrito por Markus Hutmacher
- Traducido por: Victor 2019/08/13 19:54 (UTC)

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