i3 Tiling Window Manager

i3 is a very lightweight, flexible and customisable tiling window manager with excellent multi-monitor support. Tiling window managers aim at maximising the screen space by tiling all opened windows in a non-overlapping mode. As all the bells and whistles of traditional desktops are virtually non-existent in tiling window managers, they have a very minimalist look and are mostly keyboard-focused. If the concept of tiling window managers is still puzzling, please check the official i3 screencast.

i3 in Slackware

Slackware does not come with the i3 window manager. The installation process, however, is a quick and easy one. i3 is available from SlackBuilds. Bear in mind that for due to a number of factors the latest versions of i3 will not build on slightly older versions of Slackware:

- For Slackware systems older than 14.1, you need to install i3 version 4.2.
- For Slackware 14.1, the highest i3 version that will build without replacing Slackware stock packages will be i3 4.8.
- Slackware-current (and subsequently 14.2) will be able to host the latest version of i3 (i3wm 4.11)

You can build and install the dependencies using this tutorial or via sbopkg, a SlackBuild browser.

i3 Dependencies

All i3's dependencies are also available from SlackBuilds:

- libev
- yajl
- dmenu

It is also highly recommended to install i3status and its dependency confuse.

As of version 4.8, i3 is capable of saving and restoring your layouts on particular workspaces. For more information see this.

If you'd like to utilise the new features, you need to install more dependencies. All of them are available from Slackbuilds.org:

- perl-JSON-XS
- perl-AnyEvent-Handle
- perl-AnyEvent-I3

Starting i3

Having installed all the packages, you can exit X and run `xwmconfig` to select i3 and subsequently run `startx` to start the graphical user interface.

Known Issues

i3 and nVidia Binary Driver

Up until recently, nVidia binary driver users had to add the `–force-xinerama` flag to their `.xinitrc` file.

```
# Start i3

if [ -z "$DESKTOP_SESSION" -a -x /usr/bin/ck-launch-session ]; then
    exec ck-launch-session i3 --force-xinerama
else
    exec i3
fi
```

As of version 302.17 of nVidia binary driver it is **no longer necessary**. (See more info)

i3 Configuration

When you first start i3, you will be welcomed by `i3-config-wizard`:

```
You have not configured i3 yet.
Do you want me to generate ~/.i3/config?

<Enter> Yes, generate ~/.i3/config
<Esc>   No, I will use the defaults
```

Let the wizard generate the config file. You'll then face another dilemma:

```
Please choose either:

--> <Win>  Win as default modifier
    <Alt>  Alt as default modifier

Afterwards, press

<Enter> to write ~/.i3/config
<ESC>   to abort
```

Use the Win and Alt keys to switch between the modifiers and choose one of them. In this tutorial I use Alt as the default modifier (I use Win for all sorts of custom keybindigs to avoid any clashes with i3 or other applications).

Please note that if you have changed the keyboard layout since the wizard automatically generated the config file, you might have to revisit the config file and manually modify the keybindings

Changing i3 Modifier Key(s)

It is easy to change or add i3 modifiers. Right at the top of the `~/.i3/config` file you'll see:

Alt as the default modifier:

```
set $mod Mod1
```

Or Win as the default modifier:

```
set $mod Mod4
```

You can also configure a secondary modifier assigning it to a variable (eg. `$ms`):

```
set $mod Mod1
set $ms Mod4
```

Keybindings for Most Common Activities and Applications

The following are some basic keybindings to help you get started. For a full map of default keybindings see here or consult a very thoroughly commented `config` file located in `~/.i3/`.

Terminal

Pressing Alt+Return launches a terminal which in Slackware defaults to `xterm`. If you want to change it, modify the following line:

```
bindsym $mod+Return exec i3-sensible-terminal
```

Specify a terminal of your choice:

```
bindsym $mod+Return exec /usr/bin/urxvt
```

Close a Window

Alt+Shift+Q

Go to a Given Workspace

Alt+2

In this instance we go to `Workspace 2`.

Reload the Config

Alt+Shift+C

Restart i3

Alt+Shift+R

Quit i3

Alt+Shift+E

Adding Your Own Keybindings

If you want to launch Firefox using Alt+B, add the following to `~/.i3/config`:

```
bindsym $mod+b exec /usr/bin/firefox
```

Opening Other Applications

The Alt+D keybinding launches `dmenu` where you can type a program you want to run.

Keyboard Layout

Please visit this HOWTO to configure the keyboard layout in i3.

Further Reading

i3 has a great number of features. Discussing all of them is beyond the scope of this HOWTO. For further help, please refer to i3's excellent User's Guide.

These are some notable features:

- Excellent window management (concept of container trees, a screencast on containers and the tree data sctructure)
- Automatically putting clients on specific workspaces
- The scratchpad feature
- The article describes the use of `goto` marks and Emacs-like modes in i3.

i3status

i3status is a status bar generator which will help you display all sorts of information.

Once you've started i3, you should see a status bar at the bottom of the screen. To start customising it, copy `/etc/i3status.conf` to `~/.i3status.conf` where you can place your changes. The configuration is pretty straightforward. You can comment out any modules you don't want to be displayed:

```
# order  = "ipv6"
order += "disk /"
# order += "run_watch DHCP"
# order += "run_watch VPN"
order += "wireless wlan4"
#order += "ethernet eth0"
# order += "battery 0"
# order += "cpu_temperature 0"
order += "load"
order += "time"
```

You can configure modules in the sections below. For example:

```
time {
        format = "%d-%m-%Y %H:%M"
}
```

Custom i3status Display

By default `i3status` functionality is somewhat limited. The fact that the basic configuration offers only a handful of predefined functions does not, however, prevent you from customising it to include your own scripts.

The most basic method of calling i3status is by including the following code in `~/i3.config`:

```
bar {
        status_command i3status
}
```

i3status will first look for `~/.i3status.conf` and if it is not present, it will read `/etc/i3status.conf`. You can also manually specify the location of the config file:

```
status_command i3status --config ~/.i3/scripts/i3status.conf
```

Instead of calling `i3status` here, you can run a custom script which will start `i3status`.

```
status_command /path/to/my/i3-custom-status.sh
```

i3-custom-status.sh

```
#!/bin/sh
# shell script to prepend i3status output with some custom stuff
```

```
i3status --config ~/.i3status-secondary.conf | while :
do
        read line
        LG=$(setxkbmap -print | grep xkb_symbols | awk -F"+" '{print $2}')
        pycom=$(/home/user/.i3/pys.py)
        todo=$(task ls | sed -n '4s/[[:blank:]]\+/ /pg' )
        echo "TODO:$todo | LG: $LG | $pycom | $line" || exit 1
done
```

The following should give you some idea of how you could adapt it for your own needs:

```
LG=$(setxkbmap -print | grep xkb_symbols | awk -F"+" '{print $2}')
```

The current keyboard layout is assigned to variable LG.

```
pycom=$(/home/user/.i3/pys.py)
```

The output of a Python script is assigned to variable pycon.

```
todo=$(task ls | sed -n '4s/[[:blank:]]\+/ /pg' )
```

The most important task of my todo list (TaskWarrior) is assigned to variable todo.

```
echo "TODO:$todo | LG: $LG | $pycom | $line" || exit 1
```

The contents of the variables is sent to the status bar followed by default i3status output.

i3status in a Multi-monitor Setup

You can identify your monitors using the 'xrandr' utility (please note that xrandr is not fully supported with versions older than 302.17 of nVidia binary driver):

```
$ xrandr
Screen 0: minimum 8 x 8, current 3840 x 1200, maximum 16384 x 16384
DVI-I-0 disconnected (normal left inverted right x axis y axis)
VGA-0 disconnected (normal left inverted right x axis y axis)
DVI-I-1 connected 1920x1200+1920+0 (normal left inverted right x axis y
axis) 518mm x 324mm
   1920x1200      60.0*+
...
HDMI-0 connected 1920x1200+0+0 (normal left inverted right x axis y axis)
519mm x 324mm
   1920x1200      60.0*+
...
```

The active connections are identified as DVI-I-1 and HDMI-0. Knowing this we can configure separate outputs for each display:

```
bar {
```

```
        output DVI-I-1
        status_command i3status
        font -*-terminus-bold-*-normal-*-20-*-*-*-*-*-iso8859-1
        colors {
                background: #002b36
                statusline: #586e75
                focused_workspace: $col3 $col2 $col9
                active_workspace: $col3 $col2 $col16
                inactive_workspace: $col3 $col2 $col3
                urgent_workspace: $col11 $col12 $col13
                }
}

bar {
        output HDMI-0
        status_command /home/user/.i3/scripts/i3status_script.sh
        font -*-terminus-bold-*-normal-*-20-*-*-*-*-*-iso8859-1
}
```

i3 Support

Apart from excellent documentation both for i3 users and developers, you can also get support in the following places:

- Subscribe to the mailing list (Browse archives)
- Join i3 IRC channel (#i3 on irc.twice-irc.de)
- Register with the recently created stackexchange-like FAQ section (Update: As of 20/12/2015 that site is read only. The i3-related questions have moved to a new place - see the next bullet point)
- i3wm questions on reddit

Sources

- Originally written by Marcin Herda

howtos, i3, i3status, wm, software, author sycamorex