i3

i3 is a tiling window manager. See i3 User Guide for official documentation.

Also see my notes below on various settings, modules, etc

Because this is such a broad topic, I'll put some links here for the sources I used to configure my own Manjaro Linux system running the i3wm and polybar.

Alsa / Volume Mixers - Cannot find simple element

Vim Unicode Plugin

Inserting Unicode Characters Into Vim

Polybar Module Documentation

i3-gaps

i3 has been altered for various reasons and you may want a different version, i3-gaps is a popular choice right now as it leaves a configurable amount of space between your windows that gives some visual relief to your workspace. It looks nice, depending on your opinion. To check it out, you'll need to REMOVE i3 and reinstall using an alternate version. Run the following commands -

```
sudo apt-get install software-properties-common
# Head over to https://launchpad.net/ubuntu/+ppas?name_filter=i3-gaps and pick one.
# I chose https://launchpad.net/~kgilmer/+archive/ubuntu/speed-ricer as it was recommended by
the owner / maintainer of i3 on GitHub.

# Run the following command to add the PPA to your system (DEBIAN ONLY)
#+ If you are on arch, just use yay AUR manager.
sudo add-apt-repository ppa:kgilmer/speed-rice
sudo apt update
sudo apt install i3-gaps
```

Some basic i3-gaps configurations / settings taken from My Dotfiles Repo -

```
#########################################################################
### Settings for i3-gaps  #############################################
#########################################################################

# Set inner/outer gaps default values
gaps inner 14
gaps outer -2

# Additionally, you can issue commands with the following syntax. This is useful to bind keys
to changing the gap size.
# gaps inner|outer current|all set|plus|minus <px>
# gaps inner all set 10
# gaps outer all plus 5

# Smart gaps (gaps used if only more than one container on the workspace)
smart_gaps on

# Smart borders (draw borders around container only if it is not the only container on this
workspace)
# on|no_gaps (on=always activate and no_gaps=only activate if the gap size to the edge of the
screen is 0)
smart_borders on

# Press $mod+Shift+g to enter the gap mode. Choose o or i for modifying outer/inner gaps.
Press one of + / - (in-/decrement for current workspace) or 0 (remove gaps for current
workspace). If you also press Shift with these keys, the change will be global for all
workspaces.
set $mode_gaps Gaps: (o) outer, (i) inner
set $mode_gaps_outer Outer Gaps: +|-|0 (local), Shift + +|-|0 (global)
set $mode_gaps_inner Inner Gaps: +|-|0 (local), Shift + +|-|0 (global)
bindsym $mod+Shift+g mode "$mode_gaps"

mode "$mode_gaps" {
        bindsym o      mode "$mode_gaps_outer"
        bindsym i      mode "$mode_gaps_inner"
        bindsym Return mode "default"
        bindsym Escape mode "default"
}
mode "$mode_gaps_inner" {
```

```
        bindsym plus  gaps inner current plus 5
        bindsym minus gaps inner current minus 5
        bindsym 0     gaps inner current set 0

        bindsym Shift+plus  gaps inner all plus 5
        bindsym Shift+minus gaps inner all minus 5
        bindsym Shift+0     gaps inner all set 0

        bindsym Return mode "default"
        bindsym Escape mode "default"
}
mode "$mode_gaps_outer" {
        bindsym plus  gaps outer current plus 5
        bindsym minus gaps outer current minus 5
        bindsym 0     gaps outer current set 0

        bindsym Shift+plus  gaps outer all plus 5
        bindsym Shift+minus gaps outer all minus 5
        bindsym Shift+0     gaps outer all set 0

        bindsym Return mode "default"
        bindsym Escape mode "default"
}
```

Xkeybinds

X11 can help configure media keys on laptops and aftermarket keyboards to pair with their intended use by running a command or action when pressed. This can seem confusing to configure, and may be time consuming at first but once you get the hang of it and know where to look it isn't all that bad. There is a GUI tool if you'd prefer to use it, but Ill still show how to do this via a terminal below.

```
# Install and use GUI xbindkeys-config tool on debian
sudo apt install xbindkeys-config
xbindkeys-config
# Use the GUI to set an action (command) to be performed for each key in the list
```

Through a terminal -

```
# Capture next keypress and output keycode information to console
xbindkeys --key
Press combination of keys or/and click under the window.
You can use one of the two lines after "NoCommand"
in $HOME/.xbindkeysrc to bind a key.
"(Scheme function)"
    m:0x0 + c:75
    F9

# OR

# Capture next multi-keypress and output keycode information to console
xbindkeys --multikey
Press combination of keys or/and click under the window.
You can use one of the two lines after "NoCommand"
in $HOME/.xbindkeysrc to bind a key.
Press combination of keys or/and click under the window.
You can use one of the two lines after "NoCommand"
in $HOME/.xbindkeysrc to bind a key.

--- Press "q" to stop. ---
"(Scheme function)"
    m:0x1 + c:75
    Shift + F9
# This will continue to capture until you press Q.
```

Take the above output into your clipboard and `vim ~/.xbindkeysrc` to add the commands needed. Below, I configure media keys for volume functionality -

```
#~/.xbindkeysrc
#

#Volume Up
"pactl set-sink-volume @DEFAULT_SINK@ +10%"
    m:0x0 + c:76
    F10

#Volume Down
"pactl set-sink-volume @DEFAULT_SINK@ -10%"
    m:0x0 + c:75
```

```
    F9


#Toggle Audio
"pactl set-sink-mute @DEFAULT_SINK@ toggle"
    m:0x0 + c:74
    F8
```

Thats it! Above, you could change the `pactl set-sink-mute` commands to anything youd like to happen when the F8-10 keys are pressed. After you're done, apply your changes by running `xbindkeys --poll-rc`

ArchWiki Resource

If you're having issues using certain keys, try the `xev` command. There will be a lot more output than what `xbindkeys --key` provides, but if pushing the key doesn't send output to `xev` then your system is handling the button independent from your OS.

Additionally, you can run `xbindkeys_show` to show the current settings applied with `xbindkeys`. This is useful when debugging to verify you have applied settings correctly and none are being overwritten or modified.

Backlight

run `sudo ls /sys/class/backlight` - if you see `intel_backlight` there you are in luck, follow the steps below to configure `xbacklight` to adjust your display brightness.

```
sudo apt install xbacklight
sudo vim /etc/X11/xorg.conf
# If the above file doesnt exisit, make it.
# If it does, append the lines below
Section "Device"
  Identifier  "Intel Graphics"
  Driver      "intel"
  Option      "Backlight"  "intel_backlight"
EndSection
# Save and exit, reboot your PC or logout of your xsession and login again.

# Now the below commands should work and can be bound to any key the same way we bound volume
keys in the section above
# Decrease brightness by 10%
xbacklight -dec 10
# Increase brightness by 10%
```

```
xbacklight -inc 10
```

Alternately, brightnessctl can be used to control the backlight. Run the following commands, replacing `<YOUR_USERNAME>` with the user on your system that you want to use to control backlight. For me, this was just my primary user, `kapper`.

```
git clone https://github.com/Hummer12007/brightnessctl
cd brightnessctl
sudo ./configure && sudo make install
sudo usermod -aG video <YOUR_USERNAME>
```

Then after a reboot we can run the following command to decrese brightness by 10%

```
brightnessctl s 10%-

Updated device 'intel_backlight':
Device 'intel_backlight' of class 'backlight':
        Current brightness: 14400 (15%)
        Max brightness: 96000
```

Or to increase brightness by 10%

```
brightnessctl s +10%

Updated device 'intel_backlight':
Device 'intel_backlight' of class 'backlight':
        Current brightness: 24000 (25%)
        Max brightness: 96000
```

Notification Systems

Useful commands / tools for handling desktop notification dialogs -

```
# Install, use notify-send
sudo apt install libnotify-bin
notify-send "Test Notification"

# Install kdeconnect for connecting mobile devices on the same network which have been paired
using kdeconnect-cli
```

```
sudo apt install kdeconnect
# Be sure to download the KDEconnect app on your mobile device in your respective app store
and connect to the same Wi-Fi network as your PC

# list devices with KDEconnect on your network
kdeconnect-cli -l --id-name-only
13b9d56df4c8815b KapperDroid
kdeconnect-cli -l --id-only
13b9d56df4c8815b

# Given the ID corresponding with the name you chose for your device within the KDEconmnect
app...
kdeconnect-cli --pair -d 13b9d56df4c8815b
Pair requested
# Check the KDEconnect app on your phone for the prompt, you may have to open the app and
navigate to the side panel -> 'Add new device'

# See help text
kdeconnect-cli -h
```

Polybar

Polybar is a simple community driven solution to configuring custom status bars. Generally, configurations are handled within the `~/.config/polybar/config` file, but some specific cases may require editing other files.

The general requirements of using Polybar is installation via your package manager, for me, this is `pacman`. After installing, we need to define our polybars, then configure i3 to handle these settings for us.

```
sudo pacman -Syu polybar
```

Optionally, polybar can be built from source by running the following commands. This was tested and worked for me on Ubuntu 20.04.

```
sudo apt install build-essential git cmake cmake-data pkg-config python3-sphinx python3-
packaging libuv1-dev libcairo2-dev libxcb1-dev libxcb-util0-dev libxcb-randr0-dev libxcb-
composite0-dev python3-xcbgen xcb-proto libxcb-image0-dev libxcb-ewmh-dev libxcb-icccm4-dev
libxcb-xkb-dev libxcb-xrm-dev libxcb-cursor-dev libasound2-dev libpulse-dev i3-wm libjsoncpp-
dev libmpdclient-dev libcurl4-openssl-dev libnl-genl-3-dev
```

```
git clone git@github.com:polybar/polybar.git
cd polybar
./build.sh
```

After installing, we need to configure our bars within `~/.config/polybar/config`, then we can simply run `polybar top` to run a polybar titled `top` within said config file.

Configure i3 for Polybar

To start, a default `~/.config/i3/config` will contain a block defining the `i3status` and its settings

```
bar {
        i3bar_command i3bar
        status_command i3status
        position bottom

        # please set your primary output first. Example: 'xrandr --output eDP1 --primary'
        tray_output primary
        tray_output eDP1

        bindsym button4 nop
        bindsym button5 nop
        font xft:URWGothic-Book 11
        strip_workspace_numbers yes

        colors {
            background #222D31
            statusline #F9FAF9
            separator  #454947

                                border  backgr. text
            focused_workspace  #F9FAF9 #16a085 #292F34
            active_workspace   #595B5B #353836 #FDF6E3
            inactive_workspace #595B5B #222D31 #EEE8D5
            binding_mode       #16a085 #2C2C2C #F9FAF9
            urgent_workspace   #16a085 #FDF6E3 #E5201D
        }
}
```

We are going to remove this, or comment it all out, and replace it with the `exec_always` line below. Now copy the `start-polybar.sh` script to `~/.config/polybar/` for use with i3 startup configuration

below. This is just telling i3 that we are starting Polybar from a script we've written and stored within the `~/.config/polybar/` directory on initial startup.

My `bar { ... }` define within `~/.config/i3/config` -

```
# Custom startup apps
exec_always --no-startup-id $HOME/.config/polybar/start-polybar.sh

# Don't use i3 status bar, comment out this block or remove it entirely
#bar { }
```

Now just press the `<Mod><Shift><R>` (i3 default setting) to reload i3 and your Polybars should start up instead of the default `i3status`

Define Polybars / Modules

For example, my `~/.config/polybar/config` -

```
[bar/top]
monitor = ${env:MONITOR}
width = 100%
height = 34
background = #00000000
foreground = #ccffffff
line-color = ${bar/bottom.background}
line-size = 16
spacing = 2
padding-right = 5
module-margin = 4
font-0 = NotoSans-Regular:size=8;-1
font-1 = MaterialIcons:size=10;0
font-2 = Termsynu:size=8:antialias=false;-2
font-3 = FontAwesome:size=10;0
font-4 = Unifont:size=8;0
modules-left = powermenu
modules-center = ki3
modules-right = volume wired-network clock

[bar/bottom]
monitor = ${env:MONITOR}
bottom = true
```

```
width = 100%
height = 27
background = ${bar/top.background}
foreground = ${bar/top.foreground}
line-color = ${bar/top.background}
line-size = 2
spacing = 3
padding-right = 4
module-margin-left = 0
module-margin-right = 6
font-0 = NotoSans-Regular:size=8;0
font-1 = unifont:size=6;-3
font-2 = FontAwesome:size=8;-2
font-3 = NotoSans-Regular:size=8;-1
font-4 = MaterialIcons:size=10;-1
font-5 = Termsynu:size=8:antialias=false;0
```

These first two blocks define our `top` and `bottom` status bars. Continuing on in the `~/.config/polybar/config` file, we see the defines for the modules -

```
[module/powermenu]
type = custom/menu
format-padding = 5
label-open = 
label-close = X
menu-0-0 = Terminate WM
menu-0-0-foreground = #fba922
menu-0-0-exec = bspc quit -1
menu-0-1 = Reboot
menu-0-1-foreground = #fba922
menu-0-1-exec = menu_open-1
menu-0-2 = Power off
menu-0-2-foreground = #fba922
menu-0-2-exec = menu_open-2
menu-1-0 = Cancel
menu-1-0-foreground = #fba922
menu-1-0-exec = menu_open-0
menu-1-1 = Reboot
menu-1-1-foreground = #fba922
menu-1-1-exec = sudo reboot
```

```
menu-2-0 = Power off
menu-2-0-foreground = #fba922
menu-2-0-exec = sudo poweroff
menu-2-1 = Cancel
menu-2-1-foreground = #fba922
menu-2-1-exec = menu_open-0

[module/cpu]
type = internal/cpu
interval = 0.5
format = <label> <ramp-coreload>
label = CPU
ramp-coreload-0 = ▁
ramp-coreload-0-font = 2
ramp-coreload-0-foreground = #aaff77
ramp-coreload-1 = ▂
ramp-coreload-1-font = 2
ramp-coreload-1-foreground = #aaff77
ramp-coreload-2 = ▃
ramp-coreload-2-font = 2
ramp-coreload-2-foreground = #aaff77
ramp-coreload-3 = ▄
ramp-coreload-3-font = 2
ramp-coreload-3-foreground = #aaff77
ramp-coreload-4 = ▅
ramp-coreload-4-font = 2
ramp-coreload-4-foreground = #fba922
ramp-coreload-5 = ▆
ramp-coreload-5-font = 2
ramp-coreload-5-foreground = #fba922
ramp-coreload-6 = ▇
ramp-coreload-6-font = 2
ramp-coreload-6-foreground = #ff5555
ramp-coreload-7 = █
ramp-coreload-7-font = 2
ramp-coreload-7-foreground = #ff5555

[module/clock]
type = internal/date
interval = 2
```

```
date = %%{F#999}%Y-%m-%d%%{F-}  %%{F#fff}%H:%M%%{F-}

[module/date]
type = internal/date
date =    %%{F#99}%Y-%m-%d%%{F-}  %%{F#fff}%H:%M%%{F-}
date-alt = %%{F#fff}%A, %d %B %Y  %%{F#fff}%H:%M%%{F#666}:%%{F#fba922}%S%%{F-}

[module/memory]
type = internal/memory
format = <label> <bar-used>
label = RAM
bar-used-width = 30
bar-used-foreground-0 = #aaff77
bar-used-foreground-1 = #aaff77
bar-used-foreground-2 = #fba922
bar-used-foreground-3 = #ff5555
bar-used-indicator = |
bar-used-indicator-font = 6
bar-used-indicator-foreground = #ff
bar-used-fill = —
bar-used-fill-font = 6
bar-used-empty = -
bar-used-empty-font = 6
bar-used-empty-foreground = #444444

[module/ki3]
type = internal/i3
; Only show workspaces defined on the same output as the bar
;
; Useful if you want to show monitor specific workspaces
; on different bars
;
; Default: false
pin-workspaces = true
; This will split the workspace name on ':'
; Default: false
strip-wsnumbers = true
; Sort the workspaces by index instead of the default
; sorting that groups the workspaces by output
; Default: false
```

```
index-sort = true
; Create click handler used to focus workspace
; Default: true
enable-click = false
; Create scroll handlers used to cycle workspaces
; Default: true
enable-scroll = true
; Wrap around when reaching the first/last workspace
; Default: true
wrapping-scroll = true
; Set the scroll cycle direction
; Default: true
reverse-scroll = false
; Use fuzzy (partial) matching on labels when assigning
; icons to workspaces
; Example: code;♚ will apply the icon to all workspaces
; containing 'code' in the label
; Default: false
fuzzy-match = true

[module/volume]
type = internal/alsa
speaker-mixer = IEC958
headphone-mixer = Headphone
headphone-id = 9

format-volume = <ramp-volume> <label-volume>
label-muted =  muted
label-muted-foreground = #66
ramp-volume-0 = 
ramp-volume-1 = 
ramp-volume-2 = 
ramp-volume-3 = 


[module/wired-network]
type = internal/network
interface = net0
interval = 3.0
label-connected =    %{T3}%local_ip%%{T-}
```

```
label-disconnected-foreground = #66

[module/wireless-network]
type = internal/network
interface = net1
interval = 3.0
ping-interval = 10
format-connected = <ramp-signal> <label-connected>
label-connected = %essid%
label-disconnected =    not connected
label-disconnected-foreground = #66
ramp-signal-0 =
ramp-signal-1 =
ramp-signal-2 =
ramp-signal-3 =
ramp-signal-4 =
animation-packetloss-0 =
animation-packetloss-0-foreground = #ffa64c
animation-packetloss-1 =
animation-packetloss-1-foreground = ${bar/top.foreground}
animation-packetloss-framerate = 500
```

Now that we have our status bars and Polybar Modules defined, we need to configure i3 to use Polybar instead of the default `i3status` that comes configured within the `bar { ... }` block of the i3 config file. See the beginning of this Polybar section for details on adding polybar to i3 instead, if you haven't already.

Starting Polybar

If you have one monitor, you can simply run `polybar top` to start the top status bar created above, and creating a start script should be straight-forward. If you are using multiple monitors and want to replicate the status bars across all displays, create the below script within `~/.config/polybar/`, name it what you wish, but be sure it corresponds with how you choose to `exec_always` in your i3 config later on.

```
#!/bin/bash
## Author: Shaun Reed | Contact: shaunrd0@gmail.com | URL: www.shaunreed.com ##
##  A script placed in ~/.config/polybar/ - Uses ${env:MONITOR}              ##
##  Starts polybars top and bottom on multiple displays                      ##
#################################################################################
# start-polybar.sh
```

```
# Kill any previous polybars
pkill -f polybar

# For each monitor in list up to ':'
for m in $(polybar --list-monitors | cut -d":" -f1); do
  # Reload polybars with monitor device name
  MONITOR=$m polybar --reload top &
  MONITOR=$m polybar --reload bottom &
done
```

[Polybar Startup Script Source]

Now, in your `~/.config/polybar/config` file, ensure the `${env:MONITOR}` environment variable is used to define the monitors -

```
[bar/top]
monitor = ${env:MONITOR}
width = 100%
height = 34
background = #00000000
foreground = #ccffffff
# Reduced..
```

Make the script executable and run it, polybar will start with your custom configs -

```
sudo chmod a+x start-polybar.sh
./start-polybar.sh
```

You may see errors for symbols used in fonts you do not have installed, see below for troubleshooting information.

To kill all Polybars, run `pkill -f polybar`

Verify / Install Fonts

You may run into issues with Unicode characters used in these configurations, see the links / commands below for help troubleshooting. The goal is usually to track down the font you are missing and install it, preferably via your system package manager. If you see an error like the below when starting your Polybars, this is likely the issue

```
warn: Dropping unmatched character _ (U+2581)
```

It is important to note that not defining the relevant font in the Polybar definition within `~/.config/polybar/config` will result in the same error.

Cross-check that you have the supported fonts installed by searching up your character in a Unicode Character Search and checking that a relevant font is installed with the below command

```
fc-match -s monospace:charset=04de1
```

This matches the Great Power Hexagram, which I use for my system power options / context menu.

The `fc-match` command above will output all fonts compatible with that symbol, if there is no output, see the Supporting Fonts link from the character's search result, and install it via your package manager.

If it is not installed, search fonts available to install via `pacman` package manager

```
sudo pacman -Ss ttf- |grep unicode
sudo pacman -Ss otf- |grep unicode
```

If it is installed an the error is still present, see that the corresponding font for the character is included in the define for the status bar it is used in. For example, to use the Hexagram above, I added the `Unifont:size=8;0` line to my `top` Polybar definition in `~/.config/polybar/config` -

```
[bar/top]
monitor = ${env:MONITOR}
font-0 = NotoSans-Regular:size=8;-1
font-1 = MaterialIcons:size=10;0
font-2 = Termsynu:size=8:antialias=false;-2
font-3 = FontAwesome:size=10;0
font-4 = Unifont:size=8;0
```

If still having issues, check the following commands for more info / useful output

```
# Search for installed fonts
fc-list | grep fontname
```

Arch Wiki - Fonts

Revision #23
Created 2019-09-13 08:25:09 UTC by Shaun Reed
Updated 2022-08-06 09:58:27 UTC by Shaun Reed