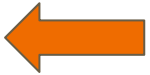
# KittyTerminalImages.jl

A package for displaying images
on the kitty terminal emulator.

# kitty - the fast, featureful, GPU based terminal emulator

- Terminal emulator like rxvt or iTerm
- Quite modern
- Fast
- Linux, Mac & Windows
- Supports images 

Download at https://sw.kovidgoyal.net/kitty/

# Escape sequences - communicate with the terminal

Special sequences of characters as commands

- Usually are of the form `"\033[ <some data> m"`
- Red Background: `"\033[30;41m"`
- Query for the text Color: `"\e]10;?\e\\"`

  => kitty answers with: `"10;rgb:d5d5/c4c4/a1a1`

# Sending an image to kitty

1. Write image to a temporary .png file.

   => Julia provides functions `mktemp()` and `tempname()`

2. Send an escape sequence with the path to kitty:

```
function draw_temp_file(path::String)
        cmd_prefix   = transcode(UInt8, "\033_Gf=100,t=t,a=T;")
        cmd_postfix  = transcode(UInt8, "\033\\")
        payload      = transcode(UInt8, base64encode(path))


        cmd = [cmd_prefix; payload; cmd_postfix]
        write(stdout, cmd)
    end
```

# Issues with this approach

1. Does not work over SSL
2. Does not work with tmux and screen

=> There might be workarounds for both problems

# Julia REPL functions for displaying Media

- For text: `show(...)`

```
julia> a = [1, 2, 3]
julia> show(a)

[1, 2, 3]
```

- For richer media: `display(...)`

```
julia> using Plots
julia> p = plot([1,2,3]);
julia> display(p)

=> opens up a new window
```

# Different representations: MIMEs

- An object can have different representations.
  Ex: Julia can show markdown  as html or latex
- Use MIME types for specifying the representation:
  `show(io::IO, m::MIME, x)`
- Create a mime object:
  `MIME"text/html"()`
- Verify if x can be shown as MIME m:
  `showable(m, x)`
- KittyTerminalImages supports MIMES "image/png",  "image/svg+xml" and "text/latex"

# Different displays: displaystack

- Media can be shown on various media:
  REPL, browser, separate window, IDE, kitty
- A display is represented by a subtype of `AbstractDisplay`
- Julia has an internal display stack: `Base.Multimedia.display`
- Packages can put their own display on this stack.
- Julia tries to display with the topmost display.

# What happens when you press enter in the REPL?

1. The line is evaluated and returns an object `x`.
2. Julia calls `display(x)`.
3. Julia selects the topmost `d::AbstractDisplay` from the display stack that knows how to display this object.
4. Julia calls `display(d, x)`.
5. Julia selects the best the best mime `m::MIME` for this display and object, and calls `display(d, m, x)`.
6. `display(d, m, x)` uses `show(m, x)` to get a text representation of `x`.

# Links

- kitty: https://sw.kovidgoyal.net/kitty
- KittyTerminalImages: https://github.com/simonschoelly/KittyTerminalImages.jl

## Other approaches:

- TerminalExtensions.jl: https://github.com/Keno/TerminalExtensions.jl
- TerminalGraphics.jl: https://github.com/m-j-w/TerminalGraphics.jl
- SixelTerm.jl: https://github.com/tshort/SixelTerm.jl