NYC Emacs Meetup Lightning Talk: Cackledaemon

Josh Holbrook

July 6 2020

whoami

- Hi I'm Josh
- I'm a data engineer
- @jfhbrook on Twitter
- @jfhbrook on GitHub

I use Org-Mode

- I have moderate-to-severe ADHD
- I need a “productivity system” to ensure I do the things I want to do
- Mine is home-grown but borrows from GTD, Agile and bullet journaling
- This is digitized using Emacs and org-mode

I use Org-Mode on multiple platforms

- I need to be able to access my projects and TODOs on all my devices
- I keep my org files in sync with Nextcloud on a NixOS DigitalOcean droplet
- I access them with beorg on iOS and Emacs on Arch Linux, OSX, and Windows

I like using Emacs in daemon mode

- In “daemon mode” Emacs runs a server in the background
- You connect to the Emacs daemon with `emacsclient`
- This means all your Emacs panels share state

Easier on some OS's than others

- Linux: systemd has your back
- OSX: a talk for another day, frankly
- Windows: kind of a problem

Windows services are not like Linux daemons

- Linux daemons are regular processes detached from a terminal
- Windows Services expose lifecycle functions instead of a `main()`

Example code from a Windows service

Via Microsoft's documentation on the same:

```
protected override void OnStart(string[] args)
{
    eventLog1.WriteEntry("In OnStart.")
}

public void OnTimer(object sender, ElapsedEventArgs args)
{
    eventLog1.WriteEntry("Monitoring the System", EventLogEn
}
```

(sorry, I don't have ob-csharp installed)

Windows does have autostart

- Windows will launch applications on login
- Those apps are then on their own
- But they are normal apps!

You can combine this with tray icons

- Tray icons are effectively “in the background” because no window
- They’re still on their own

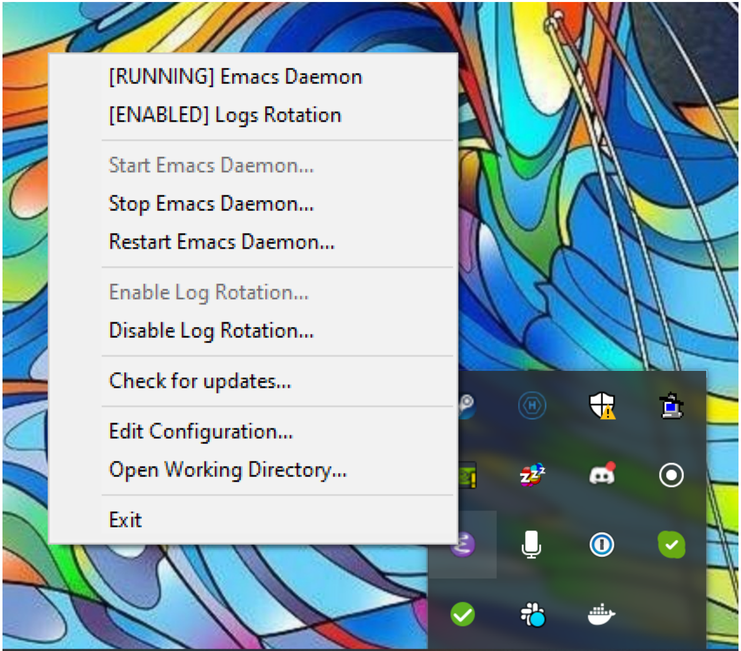
Tray icon support is included in Windows Forms

- Windows Forms is “the tcl/tk of Windows”
- It’s bundled with .NET
- It looks mad ugly but it’s easy to use

Enter: PowerShell

- PowerShell is the default shell in Windows, not Bash or ZSH
- PowerShell is typed and object-oriented
- In many ways more like Python or Ruby than Bash
- PowerShell comes with a lightweight package manager

Punchline: I wrote a tray icon app in Powershell



And I wrote it as an Org-Babel literate program

```
The Cackledaemon applet includes a menu item for checking fo
architecture of the icon combined with PowerShell's concurre
that it can't check for updates in an evented manner, but th
this item to trigger the process manually. The applet will l
wizard inside of a new PowerShell process.

#+BEGIN_SRC powershell :tangle ./Cackledaemon/Cackledaemon.p
  $InstallWizardItem = New-Object System.Windows.Forms.MenuI
  $InstallWizardItem.Text = 'Check for updates...'
  $OnInstallWizardClick = {
    Start-InstrumentedBlock 'Failed to launch install wizard
      Start-Process powershell.exe -ArgumentList @(
        '-NoExit',
        '-Command', (Join-Path $PSScriptRoot 'InstallWizard
      )
    }
  }
  $InstallWizardItem.add_Click($OnInstallWizardClick)
```

Demo Time? If there's time

I'm presenting this on my Windows machine so I can show you!

But wait there's more!

`https://github.com/jfhbrook/Cackledaemon`

Thanks