

Que Stack Gráfica Escolher?

Daniel Margarido

June 28, 2019

Índice

- 1 Problema
- 2 Estado atual
- 3 Foco de Análise
- 4 Consumos atuais
- 5 Aplicação de Notas - GKT
- 6 Aplicação de Notas - Swing
- 7 Aplicação de Notas - FLTK
- 8 Aplicação de Notas - X11 + Nuklear
- 9 Comparação de resultados
- 10 Dados de outros casos
- 11 Outras Alternativas
- 12 Conclusões
- 13 Motivação
- 14 Questões

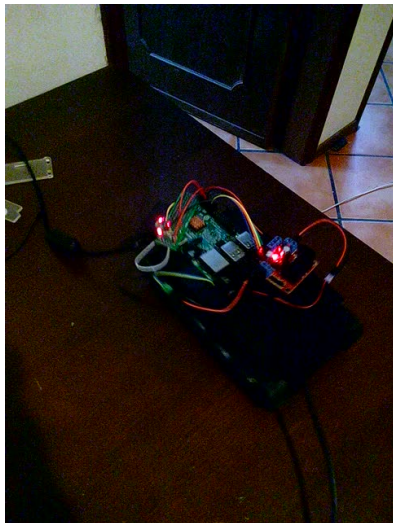
Problema

Geral

- Equipamentos ficam mais lentos ao longo do tempo.
- A verdade é que não ficam.
- O software que usamos diariamente gasta cada vez mais recursos.
- E usufruimos assim tanto de mais funcionalidades do que era usado à uns 5 ou 10 anos?

Minha Workstation

- 2 Motores 92RPM
- 1GB de RAM
- 8GB de armazenamento
- 4× ARM Cortex-A53, 1.2GHz
- Robot Tank Chasis



Estado atual

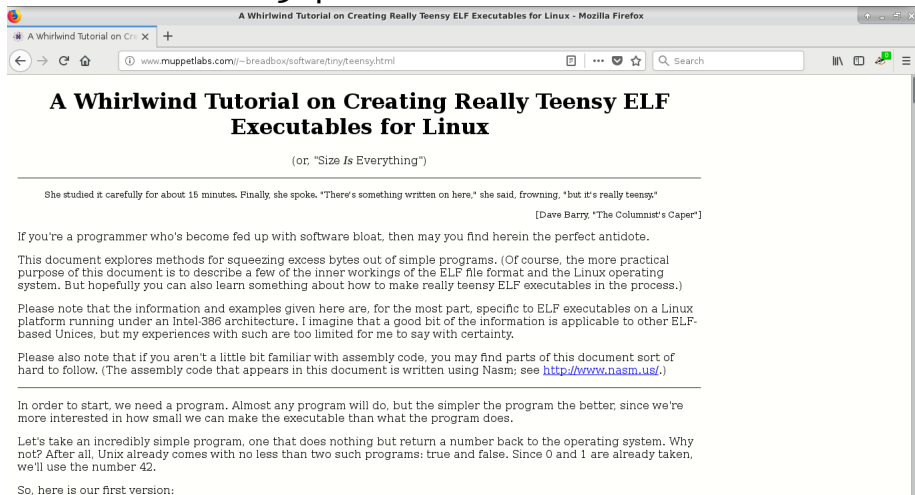
Windows 10 Acabado de Ligar

Consumo de RAM: 2.4GB



Firefox com tab de texto

Consumo de RAM: 514MB



A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux - Mozilla Firefox

A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux

(or, "Size Is Everything")

She studied it carefully for about 15 minutes. Finally, she spoke. "There's something written on here," she said, frowning, "but it's really teensy."

[Dave Barry, "The Columnist's Capers"]

If you're a programmer who's become fed up with software bloat, then may you find herein the perfect antidote.

This document explores methods for squeezing excess bytes out of simple programs. (Of course, the more practical purpose of this document is to describe a few of the inner workings of the ELF file format and the Linux operating system. But hopefully you can also learn something about how to make really teensy ELF executables in the process.)

Please note that the information and examples given here are, for the most part, specific to ELF executables on a Linux platform running under an Intel-386 architecture. I imagine that a good bit of the information is applicable to other ELF-based Unices, but my experiences with such are too limited for me to say with certainty.

Please also note that if you aren't a little bit familiar with assembly code, you may find parts of this document sort of hard to follow. (The assembly code that appears in this document is written using Nasm; see <http://www.nasm.us/>.)

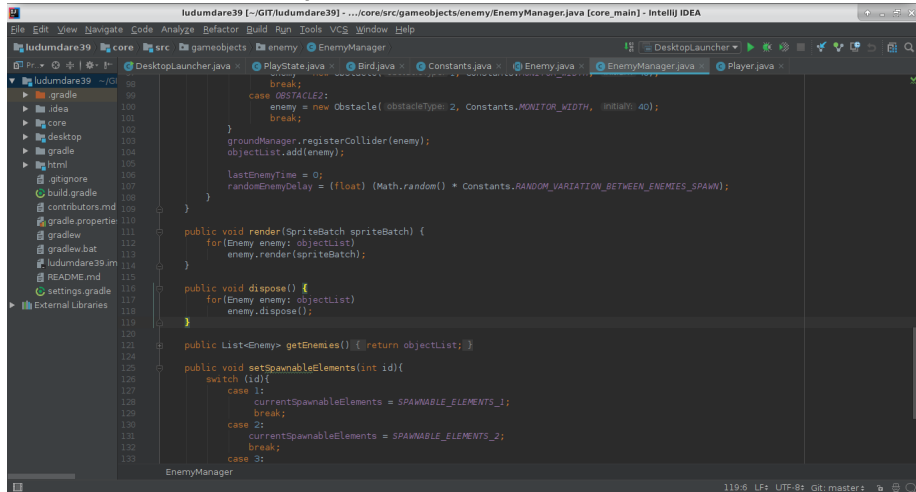
In order to start, we need a program. Almost any program will do, but the simpler the program the better, since we're more interested in how small we can make the executable than what the program does.

Let's take an incredibly simple program, one that does nothing but return a number back to the operating system. Why not? After all, Unix already comes with no less than two such programs: true and false. Since 0 and 1 are already taken, we'll use the number 42.

So, here is our first version:

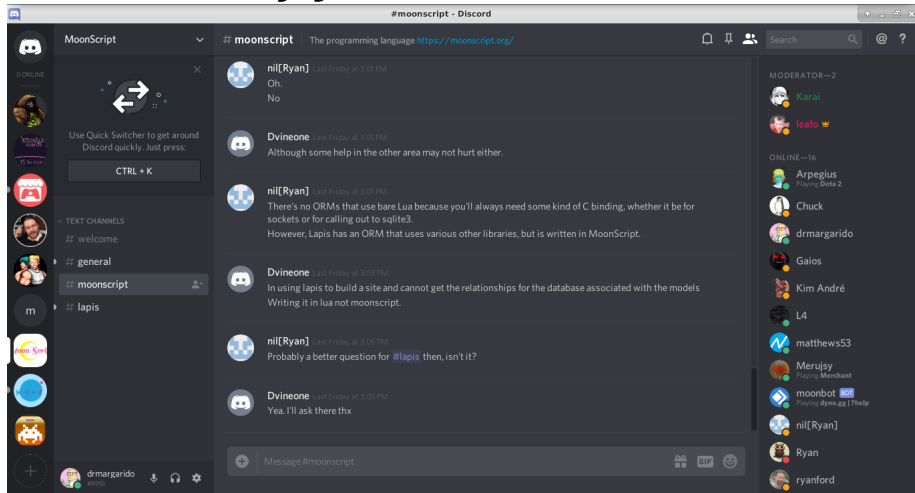
IntelliJ IDEA Acabado de Abrir

Consumo de RAM: 848MB



Discord

Consumo de RAM: 503.1MB



Foco de Análise

Recursos

- Windows ocupa para cima de 2GB e mesmo o Ubuntu atual ocupa quase 1GB de RAM.
- 20MB de RAM é quanto ocupa um servidor OpenBSD acabado de instalar.

Comparação

- Servidor que tem interface de linha de comandos vs um sistema com ambiente gráfico completo.
- 40x mais RAM, temos de tentar reduzir o máximo que conseguirmos.

Consumos Atuais

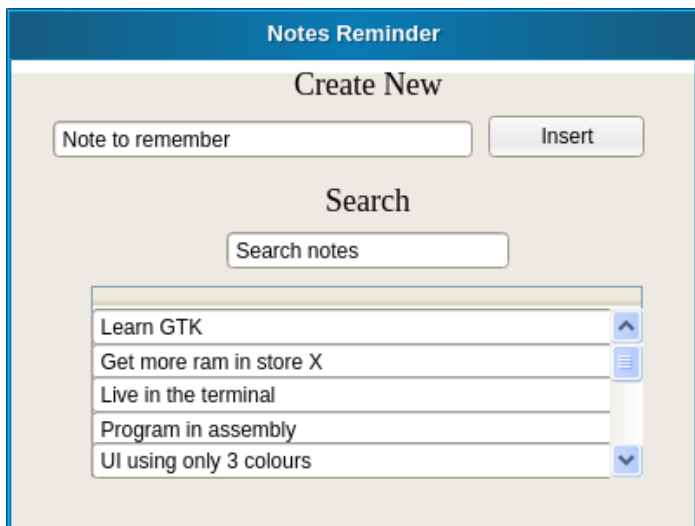
Metodologia

- Pesquisa de toolkit gráficos.
- Implementação de aplicação de notas.
- Inserção de notas no em ficheiro e a sua pesquisa.
- Em cada teste vamos medir:
 - Utilização de memória RAM
 - Simplicidade de implementação
 - Plataformas Suportadas

Regras

- Tamanho de janela 400x300.
- Implementar apenas a interface usando o toolkit gráfico.

Mockup da Aparência Desejada



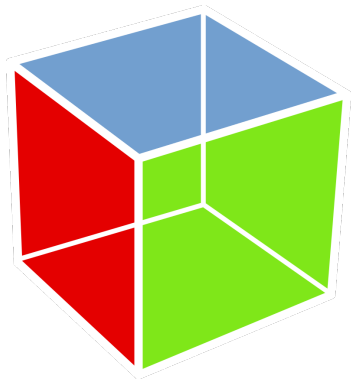
The mockup shows a window titled "Notes Reminder" with a light blue header. Below the header is a section titled "Create New" in a large, dark font. Under "Create New", there is a text input field containing the placeholder text "Note to remember" and a button labeled "Insert". Below this is a section titled "Search" in a large, dark font. Under "Search", there is a text input field containing the placeholder text "Search notes". At the bottom of the window is a list box containing five items: "Learn GTK", "Get more ram in store X", "Live in the terminal", "Program in assembly", and "UI using only 3 colours". The list box has a light yellow header and a vertical scrollbar on the right side. The list items are displayed in a white background with a thin border. The list box has a small blue button with an upward arrow on the right side and a small blue button with a downward arrow on the right side.

Figure 1: Mockup Interface

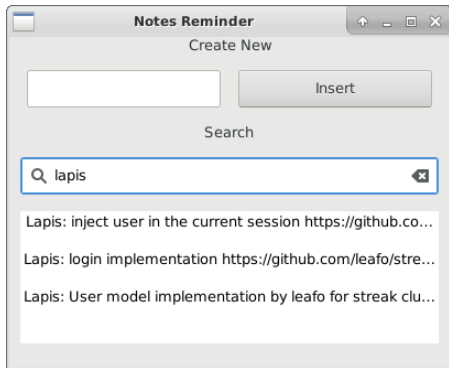
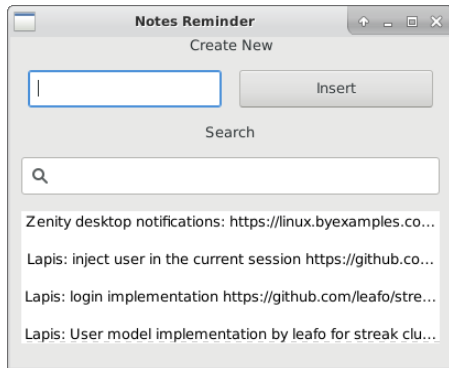
Aplicação de Notas – GKT

Apresentação

- Criado em 1998.
- Implementado em C.
- Desenvolvido pelo GNOME Project.
- Maioria dos ambientes gráficos mais utilizados em linux utilizam gtk.



Resultado



Avaliação

- Utilização de memória RAM - 26.54MB
- Plataformas Suportadas - GNU/Linux, Unix, Windows e Mac OS X
- Simplicidade de implementação:
 - Widgets baseados em GtkWidget.
 - Manual de fácil pesquisa e com boa documentação.
 - Glade para construção de interface só com drag and drop.
 - Trabalhoso usar as caixas de layout que são para definir posição dos widgets.

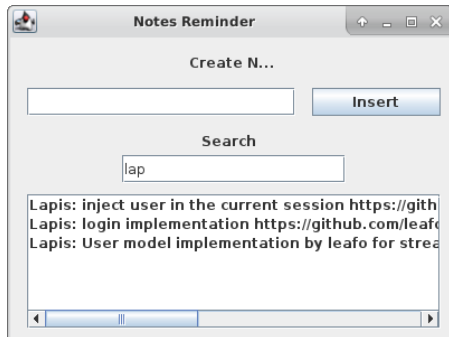
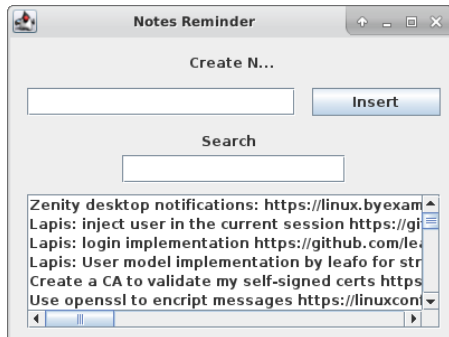
Aplicação de Notas – Swing

Apresentação

- Criado em 1997
- Implementado em Java
- Alternativa lightweight ao java AWT
- Desenha os próprios widgets sem utilizar os do sistema



Resultado



Avaliação

- Utilização de memória RAM – 55.60MB
- Plataformas Suportadas – Platform-Independent
- Simplicidade de implementação:
 - Documentação nos standards do java mas sem pesquisa rápida.
 - Escassos exemplos de utilização.
 - Utilização simples permite posicionamento usando layouts e posicionamento directo na frame.

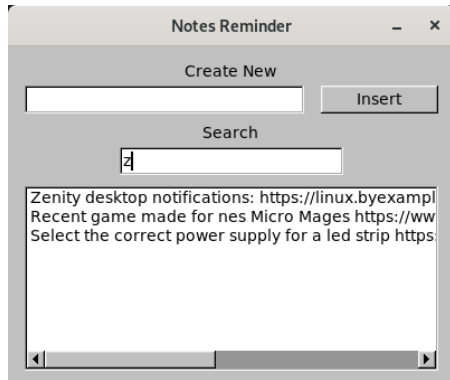
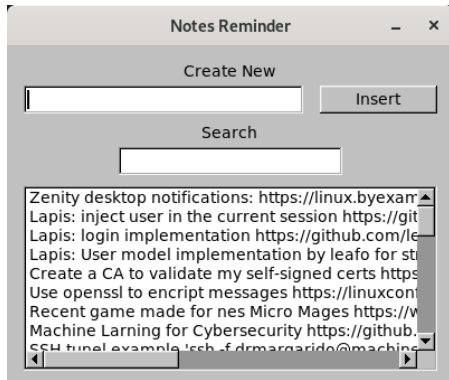
Aplicação de Notas – FLTK

Apresentação

- Criado em 1998
- Implementado em C++
- Usa um design mais leve e restringe-se apenas à funcionalidade de GUI
- Normalmente linkado de forma estática



Resultado



Avaliação

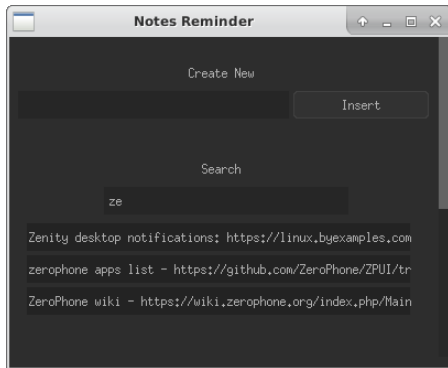
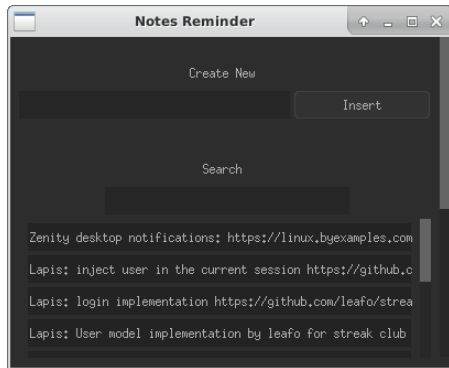
- Utilização de memória RAM - 9.84MB
- Plataformas Suportadas - GNU/Linux, Unix, Windows, macOS, AmigaOS 4
- Simplicidade de implementação:
 - Desenho de elementos na interface simples
 - Gestão de linhas e margens precisou de calculos manuais
 - Documentação tem os widgets bem documentados e aborda a maioria dos casos de uso
 - FLUID permite construir a interface com drag and drop

Aplicação de Notas – Nuklear

Apresentação

- Criado em 2015
- Implementado em ANSI C
- Immediate mode graphical user interface toolkit
- Zero dependências, recebe o input do estado atual e gera comandos de desenho de primitivas como output
- Backends atuais para desenhar estas primitivas - x11, d3d9, d3d11, gdl, gdlp, glfw, sdl com opengl, sfml com opengl, allegro5

Resultado



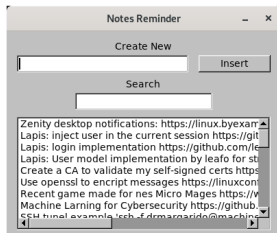
Avaliação

- Utilização de memória RAM - 3.9MB
- Plataformas Suportadas - Depende dos backends de render
- Simplicidade de implementação:
 - Utilização de contextos para desenhar demora um pouco a entender
 - Desenho por layout e posições absolutas
 - Documentação limitada
 - Utilização envolve ainda alguma leitura de código para escolher backends e perceber como é a integração

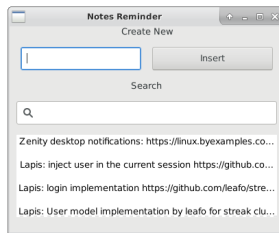
Comparação de resultados

Comparação

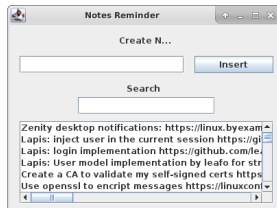
9.84MB



26.54MB



55.60MB



3.9MB

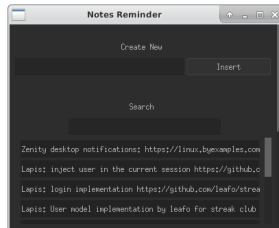


Figure 2: Toolkits Comparison

Dados de outros casos

Electron

- Hello World utiliza 125MB de RAM
- Rapidamente flutua para valores mais altos
- Chromium com aplicação no topo



Electron Example

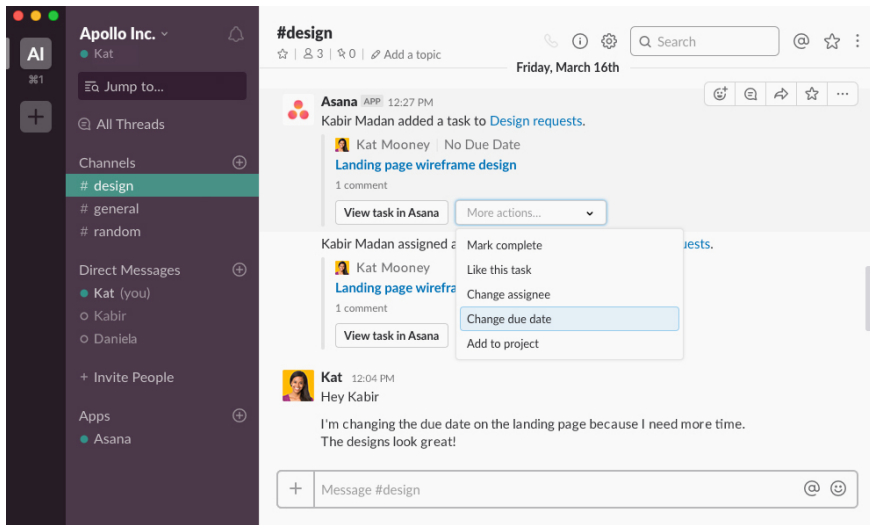


Figure 2: Slack
Que Stack Gráfica Escolher?

Tekui

- Hello World utiliza 9.8MB de RAM
- Utiliza Lua para scripting
- Permite mudar estilos com CSS
- Cross-Platform

The logo for tekUI, featuring the word "tekUI" in a green, lowercase, sans-serif font. The "i" has a dot.

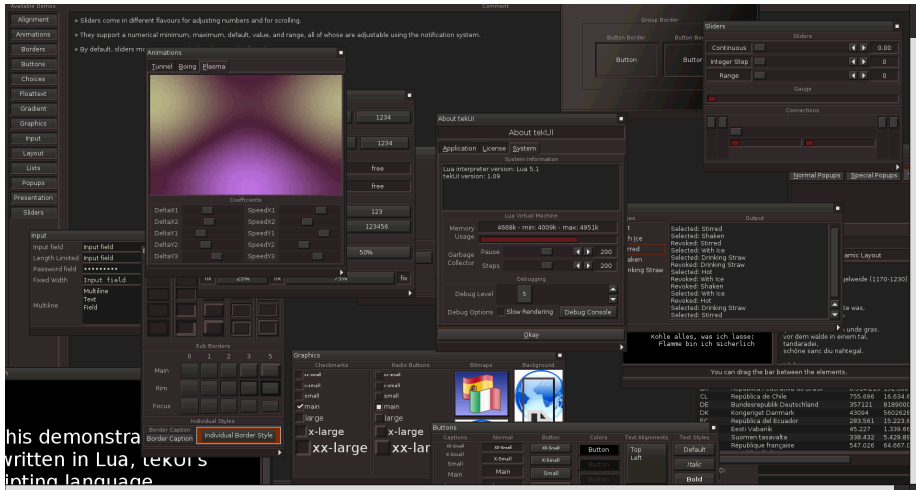


Figure 4: Tekui Usage

Wayland Client

- Hello World utiliza - 868KB de RAM
- Disponibiliza um pixelbuffer
- É preciso implementar toda a stack
- Alternativa Wayland + ImGui



Outras Alternativas para Testar

Com muitas funcionalidades

- QT
- .NET

Lightweight

- tk
- motif
- IUP
- SDL

Sistema

- Win32
- X.org

Conclusões

Conclusões

- Toolkits mais apelativos
- Maximizar a produtividade
- Cross-platform
- É possível reduzir o consumo de memória

Motivação

Futuro

- Manter o raspberry como máquina de trabalho durante mais uns 5 anos
- Não ter de trocar de hardware de 2 em 2 anos

Questões

Contactos

- Email:
 - `drmargarido@gmail.com`
- Apresentação:
 - `https://github.com/drmargarido/minimize_memory`
- Github:
 - `https://github.com/drmargarido`
- Bitbucket:
 - `https://bitbucket.org/Alface0/`
- Itch.io:
 - `https://drmargarido.itch.io/`

