

Que Stack Gráfica Escolher?

ENEI 2019

Daniel Margarido

June 7, 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 Comparação de resultados
- 9 Dados de outros casos
- 10 Outras Alternativas
- 11 Conclusões
- 12 Motivação
- 13 Questões

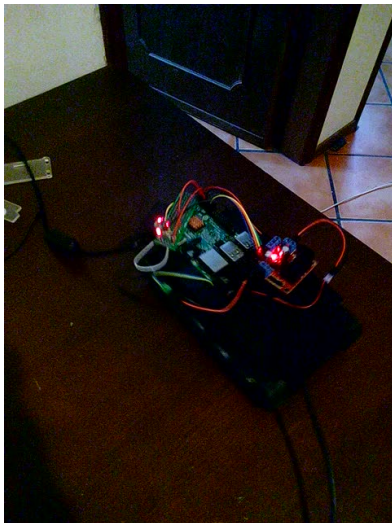
Problema

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

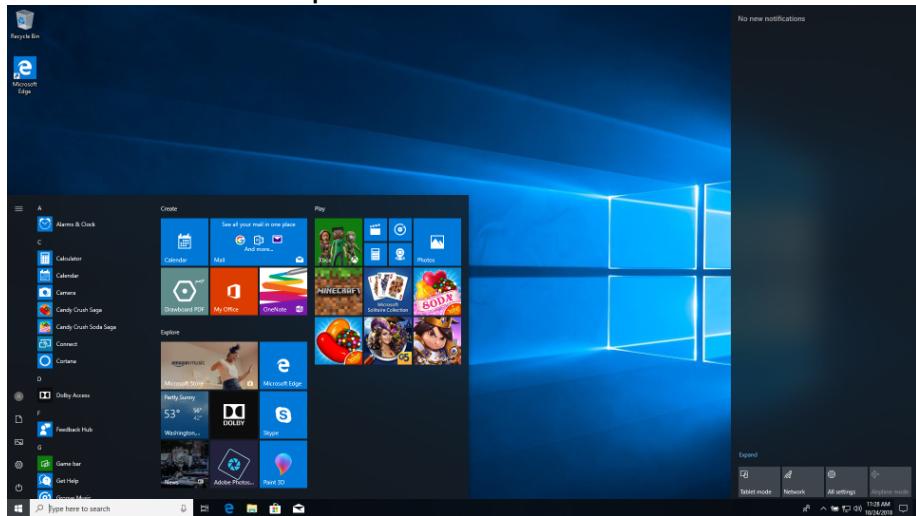
- 1GB de RAM
- 8GB de armazenamento
- 4× ARM Cortex-A53,
1.2GHz
- 2 Motores 92RPM
- Robot Tank Chasis



Estado atual

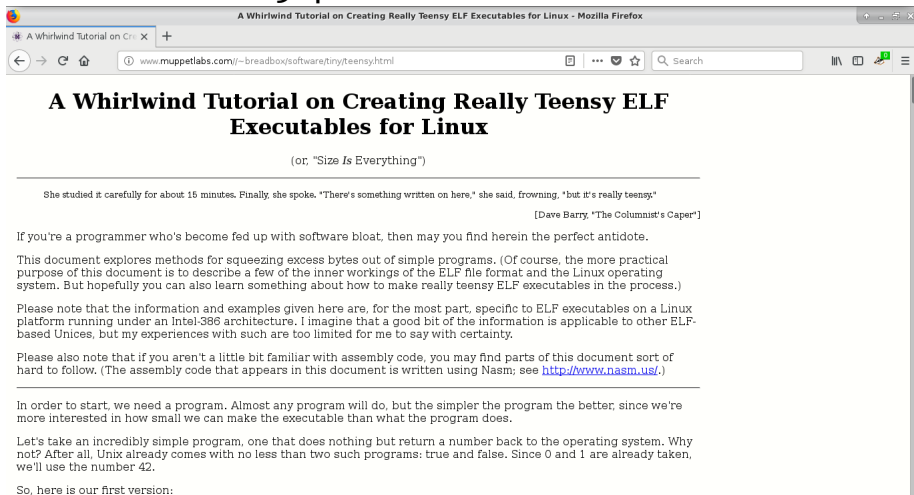
Windows 10 Acabado de Ligar

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 Cr... x +

www.muppetlabs.com/~breadbox/software/tiny/teensy.html

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 Caper"]

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/>.)

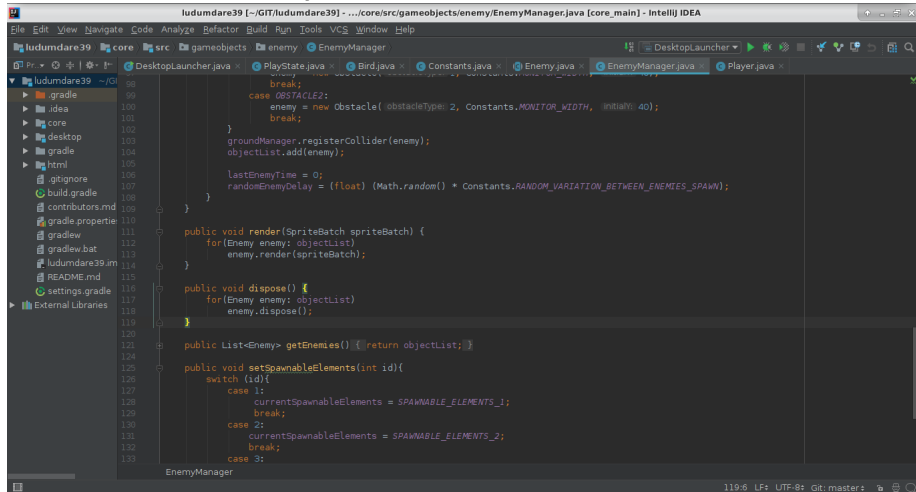
In order to start, we need a program. Almost any program will do, but the simpler the program the better; since we're more interested in how small we can make the executable than what the program does.

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



```
ludumdare39 [~/GIT/ludumdare39] - .../core/src/gameobjects/enemy/EnemyManager.java [core_main] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ludumdare39 core src gameobjects enemy EnemyManager DesktopLauncher
DesktopLauncher.java PlayState.java Bird.java Constants.java Enemy.java EnemyManager.java Player.java
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
    break;
    case OBSTACLE2:
        enemy = new Obstacle( obstacleType: 2, Constants.MONITOR_WIDTH, initialY: 40);
        break;
    }
    groundManager.registerCollider(enemy);
    objectList.add(enemy);

    lastEnemyTime = 0;
    randomEnemyDelay = (float) (Math.random() * Constants.RANDOM_VARIATION_BETWEEN_ENEMIES_SPAWN);
}

public void render(SpriteBatch spriteBatch) {
    for(Enemy enemy: objectList)
        enemy.render(spriteBatch);
}

public void dispose() {
    for(Enemy enemy: objectList)
        enemy.dispose();
}

public List<Enemy> getEnemies() { return objectList; }

public void setSpawnableElements(int id){
    switch (id){
        case 1:
            currentSpawnableElements = SPAWNABLE_ELEMENTS_1;
            break;
        case 2:
            currentSpawnableElements = SPAWNABLE_ELEMENTS_2;
            break;
        case 3:
            break;
    }
}

EnemyManager
```

119:6 LF: UTF-8: Git: master

Discord

Consumo de RAM: 503.1MB

#moonscript - Discord

MoonScript

moonscript The programming language <https://moonscript.org/>

0 ONLINE

Use Quick Switcher to get around Discord quickly. Just press: **CTRL + K**

TEXT CHANNELS

- # welcome
- # general
- # moonscript
- # lapis

dmargarido

nil[Ryan] last Friday at 3:01 PM
Oh. No.

Dvineone last Friday at 3:01 PM
Although some help in the other area may not hurt either.

nil[Ryan] last Friday at 3:01 PM
There's no ORMs that use bare Lua because you'll always need some kind of C binding, whether it be for sockets or for calling out to sqlite3. However, Lapis has an ORM that uses various other libraries, but is written in MoonScript.

Dvineone last Friday at 3:03 PM
In using lapis to build a site and cannot get the relationships for the database associated with the models Writing it in lua not moonscript.

nil[Ryan] last Friday at 3:05 PM
Probably a better question for #lapis then, isn't it?

Dvineone last Friday at 3:05 PM
Yea, I'll ask there thx

MODERATOR—2

- Karai
- leafa

ONLINE—16

- Arpegius Playing Data 2
- Chuck
- drmargarido
- Gaios
- Kim André
- L4
- matthews53
- Merujsy Playing Merchant
- moonbot Playing dyno-gg | 7help
- nil[Ryan]
- Ryan
- ryanford

Message #moonscript

Porque isto acontece?

- Empresas procuram sempre maximizar as funcionalidades que conseguem obter com o tempo dos programadores
- “O hardware é barato”
- A maioria dos programadores trabalham em sistemas relacionados com a web.
- Software com muitas funcionalidades em vez de uma filosofia minimalista.
- “Write programs that do one thing and do it well.”

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

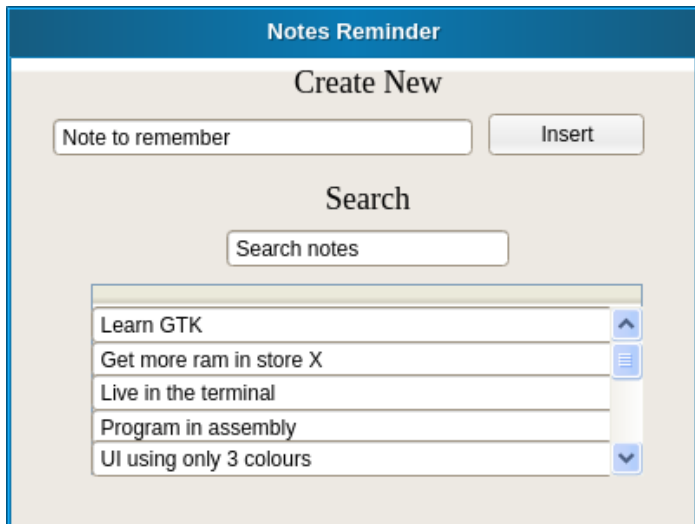
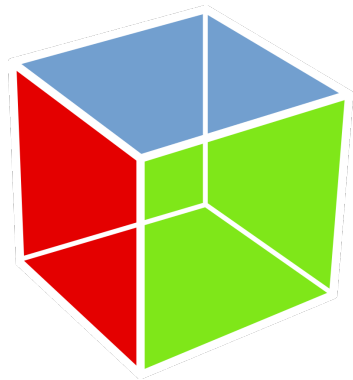


Figure 1: Mockup Interface

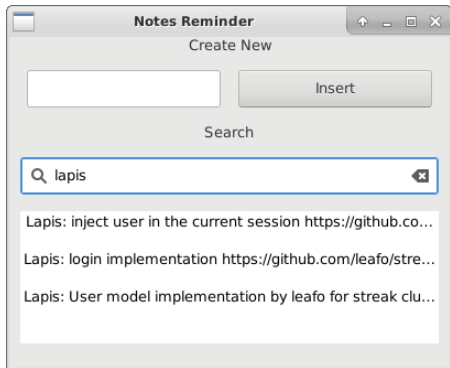
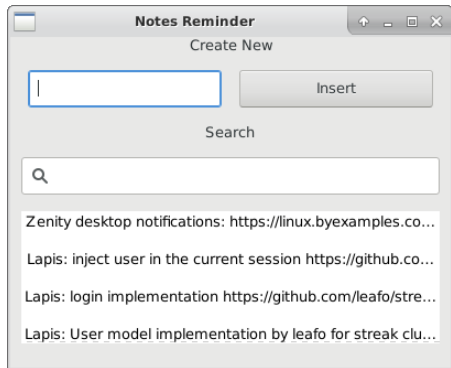
Aplicação de Notas - GKT

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.

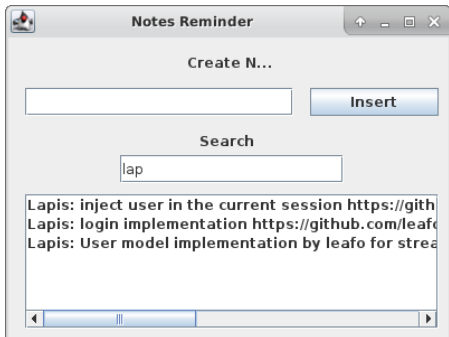
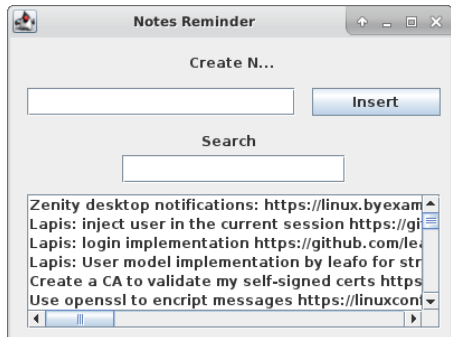
Aplicação de Notas – Swing

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.

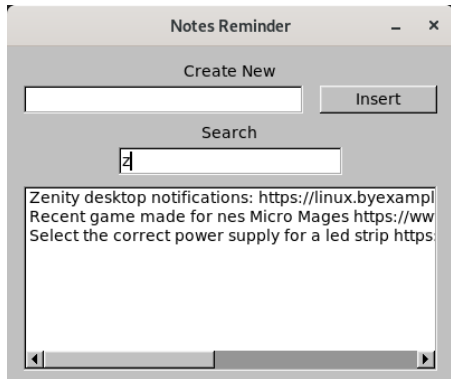
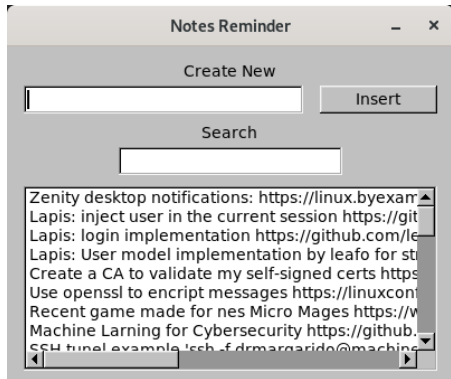
Aplicação de Notas – FLTK

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

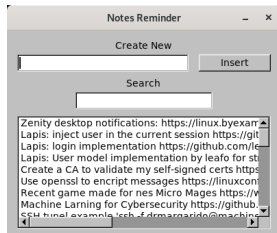
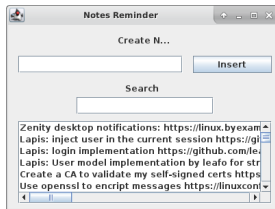
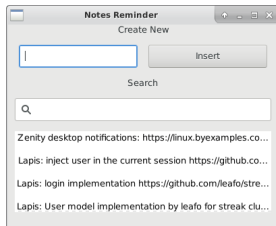


Avaliação

- 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

Comparação de resultados

Comparação



Dados de outros casos

Electron

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



Electron Example

Apollo Inc. 🔔

Kat

#1

Jump to...

All Threads

Channels +

design

general

random

Direct Messages +

Kat (you)

Kabir

Daniela

+ Invite People

Apps +

Asana

#design 👤 3 🔒 0 📌 Add a topic

Friday, March 16th

🔍 Search @ ☆ ⋮

Asana APP 12:27 PM

Kabir Madan added a task to [Design requests](#).

Kat Mooney | No Due Date

[Landing page wireframe design](#)

1 comment

[View task in Asana](#) More actions...

Kabir Madan assigned a task to Kat Mooney

[Landing page wireframe design](#)

1 comment

[View task in Asana](#)

Mark complete

Like this task

Change assignee

Change due date

Add to project

Kat 12:04 PM

Hey Kabir

I'm changing the due date on the landing page because I need more time. The designs look great!

+ Message #design @ 😊

Figure 2: Slack

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 lowercase, green, sans-serif font. The "i" has a dot, and the "U" is uppercase.

Tekui Example

Alignment

- Sliders come in different flavours for adjusting numbers and for scrolling.
- They support a numerical minimum, maximum, default, value, and range, all of whose are adjustable using the notification system.

Animations

- By default, sliders m...

Sliders

Continuous

Integer Step

Range

Gauge

Connections

Normal Poppers

Special Poppers

About tekUI

About tekUI

Application License System

System Information

Lua interpreter version: Lua 5.1
tekUI version: 1.09

Lua Virtual Machine

Memory Usage 4668k - min: 4008k - max: 4951k

Garbage Collector Pause Steps 200

Debugging

Debug Level 5

Debug Options Slow Rendering Debug Console

Output

Selected: Stirred
Selected: Shaken
Selected: With ice
Selected: Drinking Straw
Revoked: Drinking Straw
Selected: Hot
Revoked: With ice
Revoked: Shaken
Selected: With ice
Selected: Drinking Straw
Selected: Stirred

Ok

Graphics

Chadwick

Radio Buttons

Bitmaps

Buttons

Normal

Small

Medium

Large

xx-large

xx-large

xx-large

Buttons

Normal

Small

Medium

Large

xx-large

xx-large

xx-large

Text Alignment

Top

Left

Text Styles

Default

Italic

Bold

This demonstra...
written in Lua, tekui's...
programming language

Figure 3: 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

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

Software Minimalista

- suckless
- ALTERNATIVES
- harmful.cat-v
- Luke Smith
- Linux Rice

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/`

